



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y Computación

**“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA INTEGRADO
DE GESTIÓN Y CONTROL DEL TRÁNSITO VEHICULAR
BASADO EN LA UTILIZACIÓN DE SEÑALES
ELECTROMIOGRÁFICAS”**

TRABAJO DE TITULACIÓN

Previo a la obtención del Título de:

MAGISTER EN AUTOMATIZACIÓN Y CONTROL

Autores:

ALVARADO ORTIZ JIMMY EDUARDO

ROSADO ORTIZ ANTHONY JAVIER

GUAYAQUIL – ECUADOR

2019

AGRADECIMIENTOS

(Jimmy Alvarado)

Agradezco a Dios en primer lugar, por otorgarme la paciencia y sabiduría para poder adquirir la experiencia y conocimiento dentro de todo el desarrollo de esta nueva etapa académica.

A mis padres, hermana y sobrina, mi familia en general, quienes con su apoyo incondicional y su infinito amor me motivan día tras día para ser mejor persona en todo ámbito y lograr conseguir las metas tan anheladas.

A Anggie Barco, por la incalculable paciencia que ha mantenido junto a mi durante todo este trayecto académico, desde sus inicios hasta el gran final, por el inmenso amor y cariño que todos los días me demuestra.

A los amigos ganados en esta etapa: Juan Espinoza, Anthony Rosado y Paul Daza, por todo el inmenso apoyo que hemos podido brindarnos, además de compartir todo el conocimiento y experiencias técnicas y académicas.

Y un especial agradecimiento al MSc. Efrén Herrera, profesional y amigo, por la oportunidad de compartir experiencias académicas y profesionales.

AGRADECIMIENTOS

(Anthony Rosado)

Agradezco a Dios por darme la oportunidad de poder ejercer esta maestría ya que sin El, nada se puede hacer.

A mi familia que ha sido pilar fundamental, ya que su paciencia y las fuerzas que me dan me han permitido salir adelante y haber llegado a cumplir este objetivo.

A mis compañeros y amigos que hice en esta aventura llamada MACI, sin ellos no podría a ver logrado cumplir esta meta.

Gracias Jimmy Alvarado, Paul Daza, Juan Carlos Espinoza y Anthony Barahona por su paciencia y sus enseñanzas.

Un agradecimiento especial al Ing. Lenin Morejón Campoverde ya que con su conocimiento a nivel de electrónica y telecomunicaciones nos ayudó en una parte fundamental en este proyecto de titulación.

Y, por último, pero no menos importante al MSc. Efrén Herrera quien a lo largo de esta maestría se convirtió en la base principal que este proyecto de titulación se haga una realidad y siendo uno de mis formadores por sus enseñanzas, consejos y sabiduría en el ámbito académico y personal, esto me ha permitido ser una guía por el camino que en el futuro deseo tomar.

DEDICATORIA

(Jimmy Alvarado)

Para mis padres Jaime Alvarado, Cecilia Ortiz y mi hermana Gabriela Alvarado por todo y tanto, por permanecer siempre unidos, por ese amor incondicional de familia que solo nosotros conocemos, por compartir logros y fracasos juntos; a mis abuelos Jorge y Eduardo y abuelas Melba y Sara que desde el cielo guían mis pasos y me iluminan siempre; y a mi amor Angie Barco por estar siempre a mi lado compartiendo alegrías y tristezas. LO LOGRAMOS.

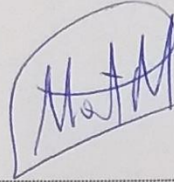
DEDICATORIA

(Anthony Rosado)

Dedicado a Dios y mi familia que son mi fuente de inspiración para superarme y cumplir mis objetivos ya sea de corto o largo plazo.

Una mención especial al Ing. Lenin Morejón Campoverde, ya que sus enseñanzas, orientación y años de conocernos, ha permitido primero establecer una linda amistad y sobre todo me ha permitido definir el camino de aprendizaje a seguir la cual actualmente me apasiona y quiero seguir aprendiendo cada día más.

TRIBUNAL DE EVALUACIÓN



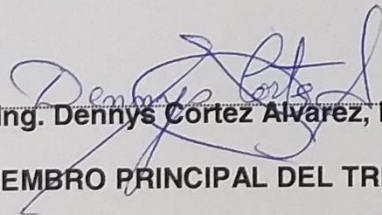
PhD. César Martín Moreno

DECANO (E) DE LA FIEC



Ing. Efrén Herrera Muentes, M. Sc.

DIRECTOR DEL TRABAJO DE TITULACIÓN

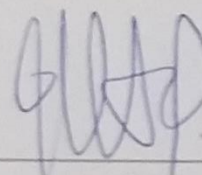


Ing. Denny Cortez Alvarez, M. Sc.

MIEMBRO PRINCIPAL DEL TRIBUNAL

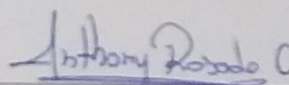
DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"



Ing. Jimmy Alvarado Ortiz

CI:0913332359



Ing. Anthony Rosado Ortiz

CI:0925163008

RESUMEN

El presente documento tiene como propósito explicar una posible solución a los problemas existentes en el control vial de las grandes ciudades que dispongan de agentes de tránsito, cuyo rol principal es el control del flujo vehicular en las grandes avenidas y calles secundarias; el gran problema es la densidad de vehículos en las denominadas “horas pico”, donde pueden generarse grandes embotellamientos. En la actualidad se están implementando sistemas denominados como “semaforización inteligente”, la cual apunta a la nueva revolución industrial 4.0. Como un aporte a esta etapa, se elabora un sistema de control de semaforización, el cual podrá ser comandado por gestos o señas que ejecuta el agente de tránsito; es decir, tendrá el control del cambio de estatus del color del mismo. En este documento se plantea una forma de transformar las señas del agente de tránsito en un lenguaje legible para el control semafórico existente. Para su realización, se plantearon 3 etapas: la primera es la adquisición de señales electromiográficas superficiales (sEMG) por medio del brazalete MYO, estableciendo como interfaz un computador de placa reducida que contiene embebido Matlab y Arduino; la segunda corresponde al procesamiento de señales mediante el uso de herramientas estadísticas, pudiendo así, identificar los sensores óptimos para el reconocimiento de patrones; y finalmente la tercera etapa que consiste en el reconocimiento de dichos patrones mediante la implementación de redes neuronales. El caso de estudio se centrará en la adquisición de los datos de una sola persona para encontrar dichos patrones y, posteriormente, implementar el control semafórico en todas sus etapas hasta llegar al sistema de semaforización existente.

ÍNDICE GENERAL

RESUMEN.....	vii
ÍNDICE GENERAL	viii
CAPÍTULO 1.....	1
1. PLANTEAMIENTO DE LA PROBLEMÁTICA.....	1
1.1 Identificación del problema	1
1.2 Justificación	2
1.3 Marco teórico.....	3
1.3.1 Señales electromiográficas superficiales (sEMG)	3
1.3.2 Adquisición y análisis de señales sEMG	4
1.3.3 Características de las señales sEMG.....	5
1.4 MYO	6
1.4.1 Estructura y características principales	6
1.5 Computadores de placa reducida	7
1.5.1 Raspberry Pi 3 B+.....	7
1.5.2 Lattepanda.....	8
1.6 El tránsito vehicular.....	9
1.6.1 Terminología.....	9
1.6.2 Modelos del flujo vehicular	9
1.6.3 Análisis de los distintos fenómenos del flujo vehicular	10
1.6.4 El agente de tránsito	10
1.7 Redes neuronales.....	11
1.7.1 Definición de redes neuronales.....	11
1.7.2 Las neuronas artificiales: esquema y características	13
1.7.3 Procesos y métodos de aprendizaje de una red neuronal.....	14
CAPÍTULO 2.....	16
2. METODOLOGÍA	16
2.1 Objetivos.....	16
2.1.1 Objetivo General	16
2.1.2 Objetivo específico.....	16

2.2 Adquisición y tratamiento de datos a ser analizados.....	17
2.2.1 Diagrama de bloque de la adquisición de datos.....	17
2.2.2 Adquisición de datos.....	17
2.2.3 Transmisión de datos.....	18
2.2.4 Recepción de datos.....	18
2.2.5 Procesamiento de datos.....	19
2.2.6 Ejecución de datos.....	19
2.3 Análisis de los datos mediante métodos estadísticos.....	20
2.3.1 Media y Mediana.....	20
2.3.2 Curtosis.....	25
2.3.3 Límites mínimos y máximos que definen la validez de un dato.....	28
2.4 Selección y validación de sensores.....	31
2.5 Análisis de la red neuronal obtenida.....	33
CAPÍTULO 3.....	40
3. RESULTADOS.....	40
3.1 Resultados obtenidos de la red neuronal aplicada.....	40
3.1.1 Resultados obtenidos de la red neuronal aplicada al primer entrenamiento... 40	40
3.1.2 Resultados obtenidos de la red neuronal aplicada al segundo entrenamiento 45	45
CONCLUSIONES.....	51
RECOMENDACIONES.....	53
BIBLIOGRAFÍA.....	54
ANEXOS.....	60
A.1 Arquitectura del proyecto.....	60
A.1.1 Funcionamiento y operatividad.....	60
A.1.2 Módulo agente.....	61
A.1.2.1 Xbee.....	61
A.1.2.2 Raspberry.....	63
A.1.2.3 Lattepanda.....	75

CAPÍTULO 1

1. PLANTEAMIENTO DE LA PROBLEMÁTICA

1.1 Identificación del problema

En la actualidad se puede afirmar que el crecimiento de la población urbana es directamente proporcional al tamaño del parque automotriz; por lo que, cada vez que se realice algún evento donde se demande la afluencia de personas, los ciudadanos saturarán las avenidas y calles.

Con el avance tecnológico existente [1], se puede indicar que los principales problemas indicados previamente se están logrando superar, y el control vial no es la excepción, ya que se encuentran desarrollando sistemas de control adaptivos e inteligentes para el manejo autónomo e integral del parque vial de las principales ciudades en el mundo: control por reconocimiento de imágenes [2], control de encendido y apagado en las luces de los semáforos adaptándolos de acuerdo a las horas pico y afluencia en ciertas avenidas y calles principales [3], preferencias a transportes de servicio público como por ejemplo ambulancias, bomberos, etc., de acuerdo a lo estipulado en la industria 4.0 tomando en consideración el internet de las cosas [4], entre otras.

Un área urbana contiene gran densidad vehicular, la cual afecta las condiciones de vida de los ciudadanos debido al tráfico que aumenta de forma alarmante; el nivel de estrés de los conductores y peatones pueden generar accidentes de tránsito, consumo excesivo de productos combustibles de origen no renovables y eleva los índices de contaminación ambiental y acústica en el sector urbano [5].

Según el Instituto Nacional de Estadísticas y Censo (INEC) en el país se encuentran matriculados 2.056.213 vehículos; de este grupo, la provincia del Guayas cuenta con 481.294 vehículos correspondientes al 23.4% del total, siendo la segunda provincia con mayor cantidad de automotores [6].

Desde el mes de julio del año 2012, el tráfico vehicular en la ciudad de Guayaquil es regulado y controlado por la Autoridad de Tránsito Municipal por medio de semáforos inteligentes, señalización y agentes de tránsito, donde la principal herramienta de tránsito utilizada es la semaforización, la cual permite el correcto flujo vehicular y peatonal.

El principal problema existente en la actualidad es el congestionamiento vehicular en la ciudad de Guayaquil donde se puede perder 27 horas al año [7]. Además, Guayaquil se encuentra dentro de las 100 ciudades donde las personas pueden pasar horas atascadas entre autos.

1.2 Justificación

La presente investigación pretende dar una solución complementaria al congestionamiento vehicular en la ciudad de Guayaquil por medio de un sistema integrado de regulación, control y gestión del tránsito, transporte terrestre y seguridad vial con el uso de semáforos por medio de señales EMG. A través de las señales EMG, las que serán proporcionadas por los agentes de tránsito, estarán en la capacidad de controlar los semáforos, con la finalidad de mejorar la fluidez vehicular, como un complemento a sus sistemas inteligentes ya existentes.

Se pretende que la Autoridad de Tránsito Municipal de Guayaquil, como un proyecto piloto, pueda implementar este proyecto, ya que en la actualidad se encuentra reinventando el tránsito con semáforos inteligentes, centros de control integrados de tránsito y transporte, paraderos inteligentes, buses eléctricos, SITU, en alianza con Google maps y waze. Lo que se aspira con este proyecto es mejorar el servicio que se ofrece en la actualidad.

Dentro del análisis y desarrollo se puede apreciar la existencia de inconvenientes: el medio en el que la muestra es tomada, debido a la existencia de señales electromagnéticas [8] de distinto origen que pueden infiltrarse como ruido, lo que desemboca en una señal no pura, tomando en consideración que las señales electromiográficas se encuentran en el orden de los milivoltios hasta los microvoltios, presentándose el ruido no deseado.

La presencia de dicho ruido afectará, en cierta manera, muchos factores como la frecuencia natural de la señal deseada, sus valores RMS, tiempos de respuesta, entre otras características de las señales EMG, dado que se presentan de manera aleatoria [9].

Las señales EMG se presentan en patrones característicos tanto en las extremidades inferiores como en las extremidades superiores de acuerdo al movimiento realizado [10]; además, debe considerarse que los patrones mencionados pueden presentar variaciones por distintos factores como el estado de ánimo de la persona, la presencia de fatiga muscular que puede generarse tanto por la toma de datos luego de haber realizado algún esfuerzo físico considerable o por la repetitividad del movimiento durante la toma de datos [11], edad, temperatura corporal [12], sexo del individuo, las cuales deberán ser consideradas como ruido a la señal pura deseada, y deberán ser filtradas.

1.3 Marco teórico

1.3.1 Señales electromiográficas superficiales (sEMG)

Las señales electromiográficas (EMG) son fuente de información apropiada para el control de dispositivos virtuales. La generación de señales EMG, se da durante la contracción muscular, donde se identifican dos estados, uno el estado transiente, lo que corresponde a los instantes de ráfaga de actividades mioeléctricas que acompaña al esfuerzo muscular repentino en la ejecución de un movimiento y dos, el estado estacionario, correspondiente al tiempo de esfuerzo muscular realizado durante un movimiento el cual es sostenido.

Las señales provenientes de los electrodos de la zona muscular son integradas y graficada en forma proporcional a los niveles de contracción y relajación del músculo, por lo que podemos decir que cuando el músculo esta tenso la gráfica se eleva y cuando el paciente relaja el músculo esta descende.

La electromiografía clínica es la metodología que se encarga del registro, estudio y análisis de la actividad eléctrica de los músculos esqueléticos, destinada al diagnóstico de posibles patologías neuromusculares, como por ejemplo el caso de la detección de enfermedades del sueño [13]. Dichas señales en conjunto con otras señales biomédicas, se complementan para diagnosticar y establecer tratamientos a enfermedades degenerativas, como lo es el Síndrome de Parkinson [14].

En la actualidad, son distintas las aplicaciones para el uso de la EMG: es utilizado en el campo clínico para el diagnóstico de complicaciones neuromusculares y neurológicos, es utilizado en el campo de la investigación, donde se encuentran involucrados la biomecánica, fisiología neuromuscular, terapia física, entre otros.

1.3.2 Adquisición y análisis de señales sEMG

La ingeniería biomédica es considerada una de las ciencias donde se incluye el estudio, desarrollo e innovación de tecnologías para fines médicos, y es dentro de esta disciplina que se encuentra el desarrollo de interfaces capaces de interpretar y adquirir las señales eléctricas correspondientes a la respuesta al estímulo al que es sometido cierto músculo del cuerpo; dichas señales son analizadas y procesadas por un dispositivo electrónico con el respectivo software [15].

En la actualidad, el desarrollo de la tecnología y la evolución de este tipo de dispositivos va en ascenso, ampliando el espectro de aplicaciones más allá de las terapéuticas y rehabilitación[16], como por ejemplo se puede citar las prótesis mioeléctricas, sino que además hacia el desarrollo y control de robots y brazos robóticos[17][18], el desarrollo de interfaces mioeléctricas para facilitar la vida diaria, inclusive para distintos usos de entretenimiento y recreación, o para reemplazar dispositivos de entrada/salida de un ordenador, como el mouse o el teclado [19].

Una de las maneras de proceder con la adquisición de este tipo de señales, son las tomadas por distintos sensores colocados sobre la superficie de la piel, considerado como una metodología no invasiva, debido a que no hay contacto directo con el músculo a ser analizado.

Existe un sinnúmero de sensores que poseen la capacidad de extraer superficialmente las señales sEMG, entre los de mayor uso se encuentran los electrodos y los dispositivos electrónicos tipo brazalete.

Los electrodos que se suelen utilizar para el análisis de este tipo de señales se clasifican en: electrodos internos y electrodos superficiales. Generalmente, los electrodos superficiales son los más acertados al momento de seleccionar el equipo para una fácil toma de muestras. Lo que garantiza el correcto envío de la señal es la colocación de un gel conductor que sirve como pegamento y adhiere el electrodo al músculo de interés, disminuyendo así la resistencia propia de la piel, acercándose a la obtención de una señal más real [20].

El sensor tipo brazalete está considerado como un sensor de tipo superficial, el cual se encarga de la captura de los impulsos nerviosos musculares de interés. Dicho equipo tuvo su origen para el control de ciertos dispositivos electrónicos actuales por medio de gestos. Por lo general, este tipo de dispositivos van ubicados alrededor del antebrazo y mediante el movimiento y activación de los músculos involucrados, además del software respectivo, se generará una acción correspondiente a un movimiento en concreto. Distintos tipos de movimiento pueden ser medibles a través de los músculos del antebrazo como un puño cerrado, mano abierta, estiramiento de cada uno de los dedos de la mano, giro de la mano, entre otros. Al ser estos equipos modernos, cuentan con su software encargado de la adquisición y análisis de datos, los cuales pueden comunicarse de manera inalámbrica hacia una PC [21][22].

1.3.3 Características de las señales sEMG

Las señales sEMG son muy complejas, aleatorias, no estacionarias, no lineales (no existe relación lineal entre la actividad muscular ejecutada y el patrón de señal de sEMG obtenido), y no son periódicas.

Para poder obtener la caracterización de las señales sEMG, es necesario la aplicación de técnicas con perspectiva en la frecuencia, en la cual se consideran completamente sus patrones: la transformada rápida de Fourier, la cual mide la amplitud de la señal sobre los componentes de frecuencia, o la frecuencia media instantánea, la que se caracteriza por presentar el valor promedio del espectro frecuencial en un instante de tiempo determinado [23].

Además, las señales sEMG son propensas a cambios generados por fatiga muscular. Cuando ocurre, la amplitud de la señal sEMG incrementa su valor, ya que los músculos intentan mantener la misma fuerza reclutando unidades motoras adicionales (esto se conoce como mecanismo de compensación de fatiga), pero cuando llega al límite, la fuerza empieza a disminuir, mostrándose una reducción en la amplitud de la señal. Adicionalmente, se presentan reducciones en la velocidad de conducción, lo que se evidencia en el cambio en las frecuencias características de la señal, en donde la acumulación del ácido láctico es una de sus principales causas[24].

De la misma manera, las unidades motoras y su frecuencia de activación, la cual se encuentra representada en la zona de bajas frecuencias del espectro frecuencial sEMG, cambia a causa de la fatiga sometida en el músculo.

1.4 MYO

1.4.1 Estructura y características principales

El MYO armband es considerado un dispositivo electrónico de ocho electrodos dispuestos en un brazalete, los cuales serán los encargados de monitorear la actividad eléctrica superficial del músculo de interés.

Su uso principalmente está destinado para el antebrazo y detecta los gestos de los músculos del miembro mencionado en conjunto con una detección de movimiento relativo, debido a que, además, dispone de mediciones inerciales de tres ejes cada uno para el acelerómetro, giroscopio y magnetómetro, convirtiendo así los gestos del músculo del antebrazo del usuario en actividades eléctricas, que pueden ser usados para distintos fines, como por ejemplo para el control de un brazo robótico [25].



Figura 1.1: MYO armband y disposición de sensores[26].

En la figura 1.1 se puede observar al MYO armband y la ubicación de los 8 sensores que componen el brazalete, además del uso en el antebrazo y la manera en que las señales del movimiento de dicho miembro pueden ser visualizadas.

Este dispositivo transmite la información requerida a través de bluetooth, pudiendo establecer comunicaciones con dispositivos compatibles. El dispositivo cuenta con una detección de 5 gestos básicos. Además, es posible acceder a las señales sEMG puras de cada uno de los sensores por medio del protocolo de bluetooth y de esta manera poder generar gestos adicionales o importantes para distintos fines.

1.5 Computadores de placa reducida

1.5.1 Raspberry Pi 3 B+

Es un ordenador de placa reducida, de bajo costo, desarrollado con el objetivo de estimular la enseñanza de informática e las escuelas dentro de Reino Unido.

Posee software de código abierto y cuenta con un sistema operativo oficial Raspbian, una versión adaptada de Debian. En todas sus versiones se incluye un procesador Bradcom, memoria RAM, GPU, puertos USB, HDMI, Ethernet, 40 pines GPIO y un conector para cámara. Ninguna edición incluye memoria, pero si un lector de tarjeta MicroSD.

La versión Raspberry Pi 3 modelo B+ aparece en marzo del 2018, actualizando a su predecesor la Raspberry Pi 3 Model B, y entre sus mejoras cuenta con un nuevo

procesador (de 1.2Ghz a 1.4Ghz) y una mejor conectividad (se incorpora doble banda a 2.4Ghz y 5Ghz y su puerto Ethernet pasa de 100Mbps a 300Mbps, y cuenta con bluetooth 4.2 de baja potencia).



Figura 1.2: Raspberry Pi 3 modelo B y modelo B+[27].

En la figura 1.2 se presenta a la izquierda la Raspberry Pi 3 modelo B y a la derecha la Raspberry Pi 3 modelo B+. La diferencia se encuentra marcada en el disipador de calor sobre el procesador y la pequeña placa con el logotipo encima del chip de conectividad inalámbrica, entre las principales.

1.5.2 Lattepanda

Es un dispositivo que tiene como objetivo mejorar los procesos robóticos e industriales ofreciendo una gama de características que no puede ofrecer las placas de Raspberry, ya que contiene un sistema operativo de Windows 10 y a su vez el manejo de Arduino, esto hace que el Lattepanda sea una herramienta de gran poder y uso.

El Lattepanda nos ofrece varios modelos, según la necesidad que se requiera. Los modelos existentes son: 2G/32GB y 4G/64GB.



Figura 1.3: Vista superior de la tarjeta LattePanda[28].

En la figura 1.3 se puede observar el modelo utilizado dentro del desarrollo de este proyecto, el estándar 2G/32GB, la cual cuenta con un procesador Intel Cherry Trail Z8300, Quad Core 1.8GHz, sistema operativo Windows 10 pre instalado, memoria ram de 2GB, capacidad de memoria de 32 GB, unidad de procesamiento gráfico Intel HD Graphics, puertos USB, conectividad por WiFi y bluetooth 4.0, salida de video tipo HDMI, soporta conectividad ethernet de 100Mbps de velocidad, contiene un procesador embebido Arduino ATmega32u4 Leonardo, 6 GPIOs para el procesador Cherry Trail y 20 GPIOs para el Arduino, alimentación de 5V/2A.

1.6 El tránsito vehicular

1.6.1 Terminología

El tránsito vehicular o también llamado tráfico vehicular, es un fenómeno originado por el flujo vehicular en una vía, similar a lo presentado en otros fenómenos como el flujo de partículas sólidas, líquidas o gaseosas.

1.6.2 Modelos del flujo vehicular

Al ser un fenómeno global, existen aproximaciones matemáticas que pretenden elaborar modelos del flujo de tránsito vehicular de acuerdo a una característica en específico. Todas ellas responden con cierto grado de apego a la realidad, midiendo una o varias características del flujo vehicular.

Los principales grupos en los que se dividen los modelos son macroscópicos, microscópicos y mesoscópicos.

Los modelos macroscópicos se enfocan principalmente en las relaciones globales del flujo vehicular tales como la velocidad de los vehículos, el flujo y la densidad del tránsito. Son por naturaleza modelos continuos, lo que implica el uso de ecuaciones diferenciales. Dentro de este grupo existen modelos netamente empíricos o modelos de capacidad y nivel de servicio [29].

Los modelos microscópicos se enfocan en la descripción del comportamiento del flujo vehicular describiendo entidades discretas individuales y su interacción entre ellas. De acuerdo a esto, son modelos discretos, por lo general [30].

Los modelos mesoscópicos o cinéticos describen una función que expresa la probabilidad de que un vehículo a determinada velocidad se encuentre en cierto tiempo en determinada posición. Utilizan por lo general métodos de mecánica estadística [31].

1.6.3 Análisis de los distintos fenómenos del flujo vehicular

Existen diversos fenómenos relacionados a ser estudiados y analizados dentro del estudio del flujo de tránsito vehicular, como, por ejemplo, accidentes de tránsito, cruces en avenidas calles de mayor tránsito, congestión de vehículos, semaforización, entre otros.

1.6.4 El agente de tránsito

El agente de tránsito es el ente encargado de regular el orden y hacer cumplir las normas de tránsito establecidas para los distintos medios de transporte. Su tarea consiste en vigilar delitos cometidos a las normas establecidas en materia de transporte, ya sea contra otros vehículos y/o transeúntes.

Además, suple las funciones de un semáforo en calles y avenidas, convirtiéndose en el director de tránsito y controlador del flujo vehicular mediante el uso de gestos que simulan la señal de “pare” equivalente al estatus rojo del semáforo y la señal de “siga”

equivalente al estatus verde del semáforo.

1.7 Redes neuronales

1.7.1 Definición de redes neuronales

Las redes neuronales artificiales son modelos computacionales que se inspiran en el comportamiento de las neuronas biológicas. Consiste en un conjunto de unidades llamadas neuronas artificiales, las cuales se encuentran conectadas entre sí para transmitirse información. Los datos o información que ingresa a la red neuronal se encuentran sometidos a una serie de operaciones internas produciendo distintos valores de salida.

Su objetivo principal es resolver problemas al igual que lo ejecutaría el cerebro humano, aunque estas redes son más abstractas. Se han utilizado para la resolución de una amplia gama de tareas, como el reconocimiento de voz [32] o la visión computarizada[33], las cuales contienen una compleja resolución que no podría ejecutarla la programación convencional basada en reglas.

Todo inició con el modelo de lógica umbral, modelo informático para redes neuronales creado por Warren McCulloch y Walter Pitts, basado en matemáticas y algoritmos. Dicho modelo trazó el camino para que la investigación de las redes neuronales se divida en dos enfoques: uno se enfocó básicamente en las funciones de su homólogo biológico, es decir, se centró en los procesos biológicos propios del cerebro[34], y el otro se enfoca en su aplicación para la inteligencia artificial[35].

Para una mejor apreciación de la operatividad y funcionamiento de una red neuronal, primero se debe conocer el funcionamiento de una neurona biológica.

La neurona es una célula componente principal del sistema nervioso, cuya función principal es la de recibir, procesar y transmitir información por medio de señales químicas y eléctricas gracias a excitabilidad eléctrica de su membrana plasmática. Se encuentra compuesta por las siguientes partes: cuerpo celular o soma, una o varias prolongaciones cortas encargadas de transmitir los impulsos hacia el soma llamadas dendritas, y una prolongación larga que conduce los impulsos desde el soma hacia otra neurona, llamada axón, tal como se muestra en la figura 1.4. La conexión

realizada entre las neuronas se conoce como sinapsis.

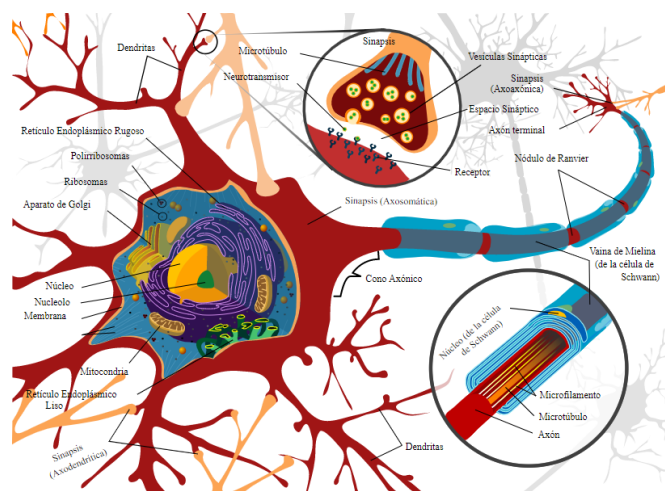


Figura 1.4: Partes de una neurona[36].

La sinapsis es una aproximación intercelular existente entre neuronas, donde se lleva a cabo la transmisión de un impulso nervioso. Se inicia con una descarga química que origina una corriente en la membrana de una célula presináptica; cuando dicho impulso alcanza el extremo del axón, se segregan compuestos químicos que son depositados en el espacio sináptico, los cuales son los encargados de excitar o inhibir la otra célula post sináptica.

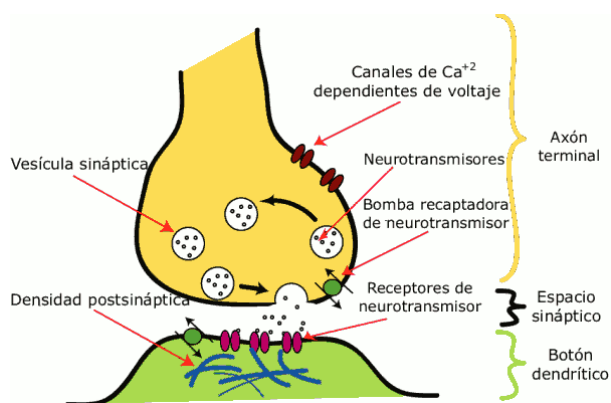


Figura 1.5: Esquema de una sinapsis[37].

En la figura 1.5 se puede apreciar el esquema de una sinapsis; se observa como la

comunicación existe entre las células nerviosas mediante la transformación de la señal eléctrica en química a través de los axones y dendritas.

1.7.2 Las neuronas artificiales: esquema y características

El modelo básico mediante el cual se encuentra diseñada una neurona artificial, siempre intenta aproximarse a su homólogo biológico, de acuerdo a lo mostrado en la figura 1.6.

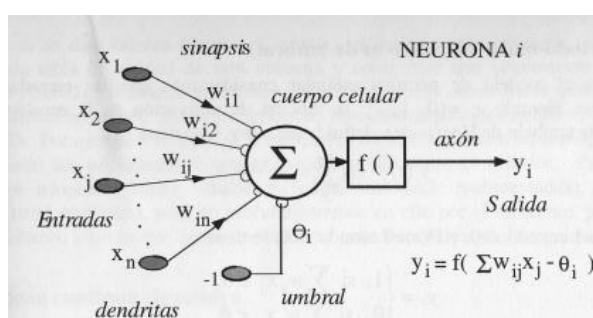


Figura 1.6: Estructura standard de una neurona artificial[38].

En la figura 1.6 se puede observar la composición o estructura standard de una neurona artificial. Dicha estructura presenta un vector de entrada x ($x_1, x_2, \dots, x_j, \dots, x_n$) mediante el cual se pretende obtener una salida única. Este vector de entrada puede homologarse como las dendritas de la neurona.

Posterior a este elemento, se encuentra la variable w , la cual es la encargada de otorgar el peso correspondiente al elemento del vector de entrada para ser enviada a la siguiente capa. Estos pesos van a ir modificándose durante cada entrenamiento de la red neuronal, para así obtener la salida deseada. Esta etapa puede considerarse como la sinapsis de su homólogo biológico.

Luego, llegamos al cuerpo celular, o la etapa de procesamiento de la información, donde llegan todas las entradas con sus pesos correspondientes, donde se ejecuta la suma ponderada.

Finalmente se llega a la etapa donde se deberá identificar la función de activación. Dicha función es la encargada de mostrar el valor contenido de acuerdo a la regla

existente para la propagación. Las funciones de activación que comúnmente son más utilizadas son las que se muestran en la siguiente Tabla 1:

Tipo	Función	Rango
Identidad	$y = x$	$[-\infty, +\infty]$
Escalón	$y = \text{sign}(x)$	$\{-1, +1\}$
Lineal	$y = \begin{cases} -1, & x < -1 \\ x, & -1 \leq x \leq 1 \\ +1, & x > 1 \end{cases}$	$[-1, +1]$
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$	$[0, +1]$
Sinusoidal	$y = A \sin(wx + \varphi)$	$[-1, +1]$
Gaussiana	$y = A e^{-Bx^2}$	$[0, +1]$

Tabla 1: Funciones de activación para modelos de redes neuronales.

La Tabla 1 muestra las principales funciones de activación para los distintos modelos de redes neuronales.

La red neuronal se compone de tres capas, la primera capa denominada como capa de entrada, la cual contiene toda la información que será procesada, la segunda capa o capa oculta, la cual es la capa de procesamiento de la información proveniente de la capa de entrada, y por último la capa de salida, la cual contiene la información o el valor esperado[39].

1.7.3 Procesos y métodos de aprendizaje de una red neuronal

El factor primordial para el aprendizaje de una red neuronal, es la cantidad de datos existentes y en conjunto con el número de iteraciones que los mismos proporciona a la red.

El aprendizaje es el proceso mediante el cual, una red modifica el valor de sus pesos en respuesta al vector de información de entrada. Los cambios que se generan durante esta etapa de aprendizaje se reducen a la destrucción (cuando el peso de la

conexión es cero), modificación y creación (cuando el valor del peso es distinto a cero) de conexiones neuronales.

Se considera que un proceso de aprendizaje ha concluido cuando los valores de los pesos permanecen estables. Es por esta razón que la manera en que se van a modificar los pesos comprende un criterio importante para determinar la regla de aprendizaje de la red.

Se consideran dos tipos de reglas: el aprendizaje supervisado y el aprendizaje no supervisado. La diferencia que caracteriza cada regla, radica en la presencia o no de un agente externo que controle el proceso.

Un criterio adicional se basa en considerar si la red puede realizar su aprendizaje durante su funcionamiento (aprendizaje online) [40] o si es necesaria una etapa previa de entrenamiento (aprendizaje offline) [41]. Para el aprendizaje offline, debe existir un conjunto de datos de entrenamiento y un conjunto de datos de prueba; de la misma manera los pesos de las conexiones no deberán ser modificados luego de finalizada la etapa de entrenamiento de la red [42]. De manera distinta ocurre en el aprendizaje online, ya que los pesos varían dinámicamente cada vez que se presente una nueva información en el sistema [43].

CAPÍTULO 2

2. METODOLOGÍA

2.1 Objetivos

2.1.1 Objetivo General

Diseñar un controlador que nos permita operar a distancia un control semafórico existente en campo mediante el monitoreo y adquisición de las señales sEMG provenientes de la extremidad superior del cuerpo humano.

2.1.2 Objetivo específico

- Obtener la caracterización de la red neuronal mediante el uso del banco de datos adquiridos en el muestreo de señales, con la finalidad de minimizar el error de salida esperada.
- Mejorar la información del tráfico vehicular, mediante la inclusión de nueva señal luminosa para la visualización de los conductores.
- Desarrollar un controlador que permita interpretar las señales sEMG a una señal más adecuada y entendible para el semáforo para ejercer el cambio de señal de luces de acuerdo al movimiento que emita el agente de tránsito.

2.2 Adquisición y tratamiento de datos a ser analizados

2.2.1 Diagrama de bloque de la adquisición de datos

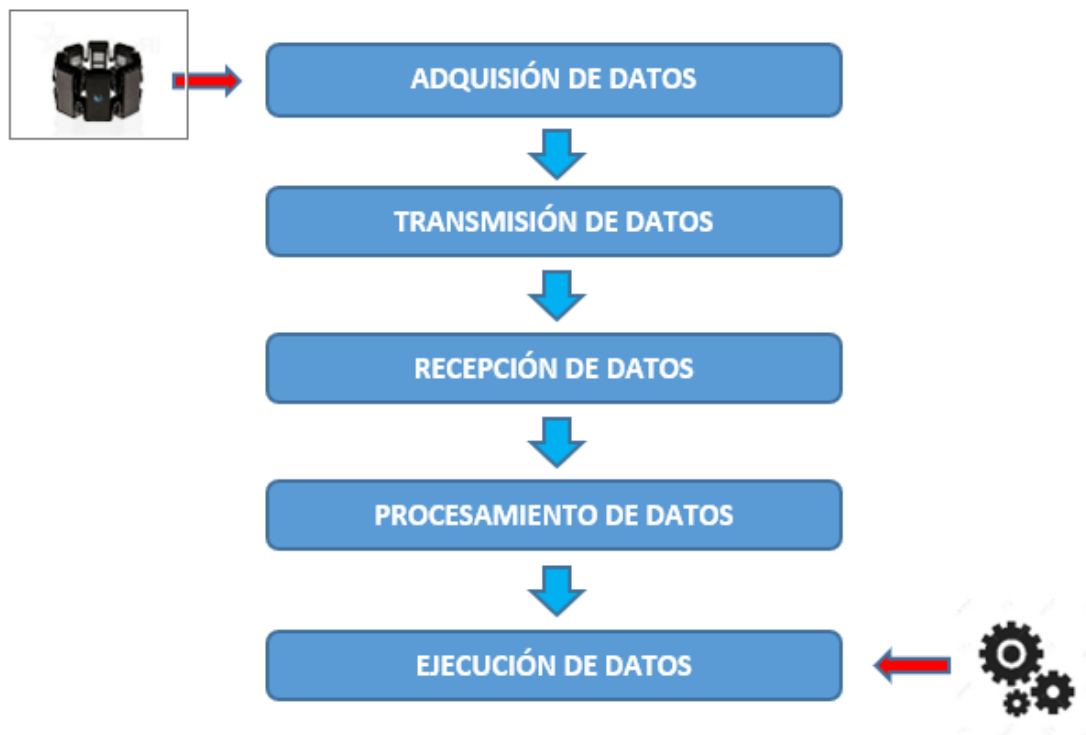


Figura 2.1: Diagrama de bloque de adquisición de datos.

En la figura 2.1 se observa el diagrama de bloque para la adquisición de datos, además de las distintas etapas en las que se compone el proyecto.

2.2.2 Adquisición de datos

Las señales de los ocho sensores que componen la MYO, son adquiridas, en primera instancia, por el dispositivo Raspberry, el cual contiene una programación elaborada en Python. Dicho dispositivo posee una interfaz gráfica, mediante la cual, el agente de tránsito selecciona el comando a ejecutarse dentro de la programación de adquisición de datos.

Luego de que la solicitud de adquisición de datos fue realizada, el dispositivo Raspberry adquiere los datos de los 8 sensores de la MYO, generando una matriz de

datos de [90x8], donde la columna de la matriz indica la posición del sensor dentro de la MYO, mientras que la fila de la matriz, indica la cantidad de datos adquiridos.

2.2.3 Transmisión de datos

Esta etapa del programa se caracteriza por seleccionar los datos que se catalogarán como válidos para ser analizados posteriormente.

Dicha selección se ejecuta mediante el comando de SOLICITUD, siempre que se encuentre inicialmente dentro de la opción de MODO AGENTE.

Al seleccionar el comando de SOLICITUD en la interfaz gráfica, el programa se encarga de, inicialmente, limpiar el buffer de datos; luego se ejecuta el indicador auditivo inicial (sonido de silbato similar al utilizado por el agente de tránsito) y en ese instante, el agente procede a ejecutar el movimiento deseado. Los datos serán capturados hasta el instante en que el indicador auditivo final (el cual es el mismo sonido de la inicial), el cual contiene todos los datos generados y capturados de cada uno de los ocho sensores que componen la MYO.

Esta manera de ejecutar la selección y captura de datos, resulta efectiva debido a que, el dispositivo Raspberry ejecuta dicha acción en un intervalo de tiempo definido, evitando así datos erróneos que puedan haberse generado por algún movimiento innecesario.

Luego de los datos generados son los correctos esta es enviada por comunicación de puerto serial utilizando un dispositivo electrónico adicional llamado Xbee, la cual nos permite enviar información vía radio a distancia muy largas.

2.2.4 Recepción de datos

Para la recepción de los datos se utiliza nuevamente un dispositivo electrónico Xbee la cual será conectado mediante un puerto de comunicación de tipo serial.

Los datos recibidos por el puerto de comunicación serial son capturados mediante un sistema realizado en .NET la cual contiene dos funciones fundamentales.

La primera es de guardar toda la información recibida en un archivo .csv y la segunda es de indicar a Matlab mediante una librería del fabricante cual es la ruta y el nombre del archivo que se generó y realice el proceso respectivo de reconocimiento de movimientos mediante la red neuronal y datos estadísticos.

2.2.5 Procesamiento de datos

Debido a que los datos generados por los sensores ubicados en la MYO, se visualizan valores tanto positivos como negativos, se decide trabajar con sus valores absolutos para poder así obtener el área bajo la curva, el cual se convertirá en el valor característico para proceder con el análisis de cada uno de los sensores. La unidad de medida correspondiente a los datos generados por los sensores es [MAV], o Mean Absolute Value por sus siglas en inglés, que significa valor medio absoluto.

Cabe indicar que, inicialmente, se plantea la normalización de las distintas curvas; sin embargo, durante las pruebas realizadas, no resulta un parámetro válido debido a que dichos valores podrán variar de acuerdo a la intensidad del movimiento ejecutada por la persona, lo que provocaría un error de lectura sobre el sensor y, así mismo, un error en el análisis de datos respectivos.

2.2.6 Ejecución de datos

Luego de realizar el procesamiento de los datos es importante obtener un valor que represente toda la cantidad de datos obtenidos sobre un sensor. De esta manera, se recurre al cálculo del área bajo la curva de cada una de las señales de los ocho sensores y de esta manera, representar un movimiento específico mediante un único valor numérico. Para el cálculo del área bajo la curva, se ejecuta la regla del trapecio, la cual consiste en dividir una curva en pequeñas secciones, que tenderán a aproximarse a la función original. El comando utilizado dentro de la programación de Matlab es el comando "trapz".

Además, para ejecutar las funcionalidades de redes neuronales se creó una función en Matlab, la cual será invocada por la librería de Matlab en el proyecto de .NET.

2.3 Análisis de los datos mediante métodos estadísticos

Para el análisis dentro de este proyecto, se toman 60 muestras por señal, es decir, 60 muestras correspondientes a la señal de “pare” y 60 muestras correspondientes a la señal de “siga”.

Cabe destacar que este muestreo fue ejecutado sobre un único individuo, debido a que las señales electromiográficas se ven afectadas por distintos factores: edad, género, índice de masa muscular, velocidad en ejecución, temperatura corporal [44], temperatura ambiente, entre otras.

2.3.1 Media y Mediana

Todo conjunto de datos tiene al menos dos características principales: centro y dispersión; y de acuerdo a la distribución y comportamiento de las características mencionadas, los datos a estudiar se encuentran agrupados dentro del grupo conocido como tendencia central.

Los datos estadísticos de tendencia central son los que se ubican al centro de la distribución de los datos, siendo los principales de estudio dentro de este proyecto la media aritmética y la mediana. Se descarta el estudio de la moda debido a que, dentro del análisis de los distintos grupos de datos, la obtención de dicho valor no fue satisfactoria debido a la ausencia del mismo.

En primera instancia, se procede con el análisis de los valores obtenidos tanto en la media aritmética como en la mediana y así observar las características de la distribución de los datos.

Sensor	Media [MAV]	Mediana [MAV]
S1	3682,742	3093,50
S2	5377,2583	4491,50
S3	7449,525	6627,50
S4	9083,7417	8682,50
S5	12889,733	12664,00
S6	6403,13	5592,50
S7	5102,1583	4820,75
S8	3127,825	2930,00

Tabla 2: Comparación estadística de media aritmética y mediana para la señal de “siga”.

En la tabla 2 se muestran los valores de las medias y medianas para cada uno de los sensores que componen la MYO.

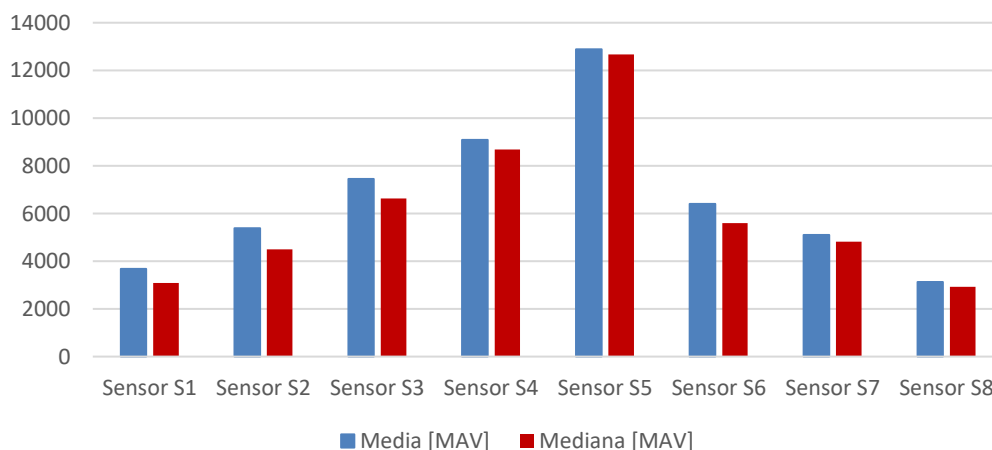


Figura 2.2: Comparación estadística de media y mediana de los sensores para la señal de “siga”.

Por medio del gráfico de barras de la figura 2.2 se puede observar que, para la señal de “siga”, ninguno de los valores de media (color azul) y mediana (color rojo) para

cada uno de los sensores son iguales, lo que indica que la distribución de datos de cada uno de los sensores no es simétrica. Además, el valor de media supera el valor de la mediana para todos los sensores S1, S2, S3, S4, S5, S6, S7 y S8, lo que indica que su distribución es asimétrica sesgada hacia la derecha.

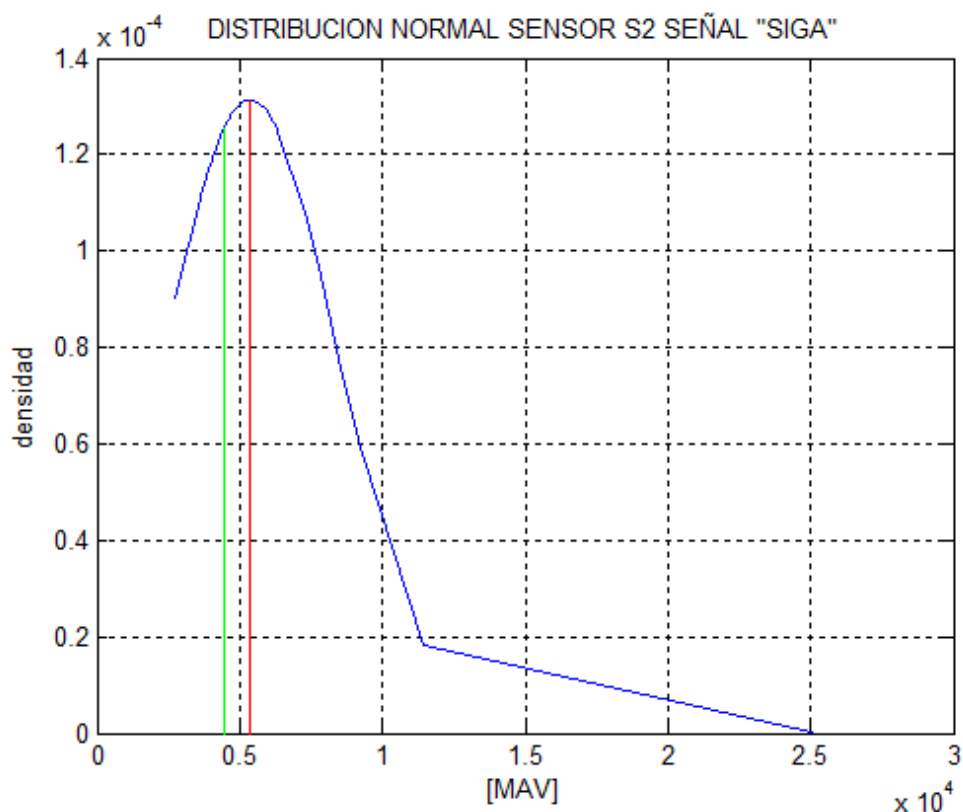


Figura 2.3: Gráfica de distribución normal sensor S2 señal "SIGA".

En la figura 2.3 se observa la gráfica de la distribución normal para el sensor S2 correspondiente a los datos obtenidos de la señal "SIGA", en color azul. La curva de color rojo corresponde a la media, mientras que la curva de color verde corresponde a la mediana. El valor de la media es mayor al valor de la mediana, por esta razón, su gráfica presenta un sesgo hacia la derecha. El sesgo en la distribución asimétrica indica que la densidad más considerable de datos dentro de la distribución se ubica hacia un lado de la mediana, llegando a considerar los datos ubicados en la cola del sesgo como valores extraños o despreciables.

Sensor	Media [MAV]	Mediana [MAV]
S1	5053,275	4855,25
S2	6117,1	5620,00
S3	11534,192	10817,25
S4	37375,008	35819,50
S5	75147,95	75590,00
S6	35624,625	34874,25
S7	37175,03	36010,25
S8	7515,175	6981,25

Tabla 3: Comparación estadística de media aritmética y mediana para la señal de “pare”.

En la tabla 3 se muestran los valores de las medias y medianas para cada uno de los sensores que componen la MYO.

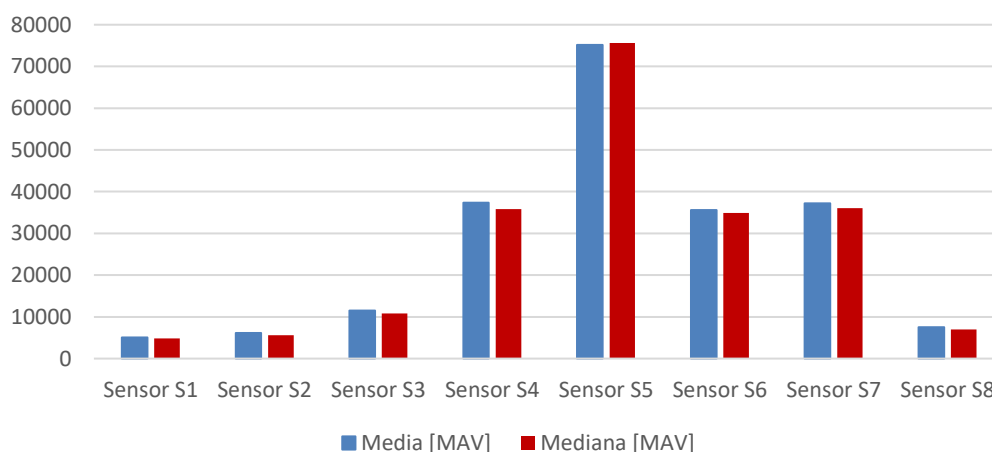


Figura 2.4: Comparación estadística de media y mediana de los sensores para la señal de “pare”.

Por medio del gráfico de barras de la figura 2.4 se puede observar que, para la señal de “pare” de los sensores S1, S2, S3, S5, S6, S7 y S8, el valor de media (color azul)

supera el valor de la mediana (color rojo), lo que indica que la distribución de datos de cada uno de los sensores es asimétrica sesgada hacia la derecha. Sin embargo, para el sensor S5, el valor de la mediana es aproximadamente igual al valor de la media, por lo que se asume como una distribución simétrica.

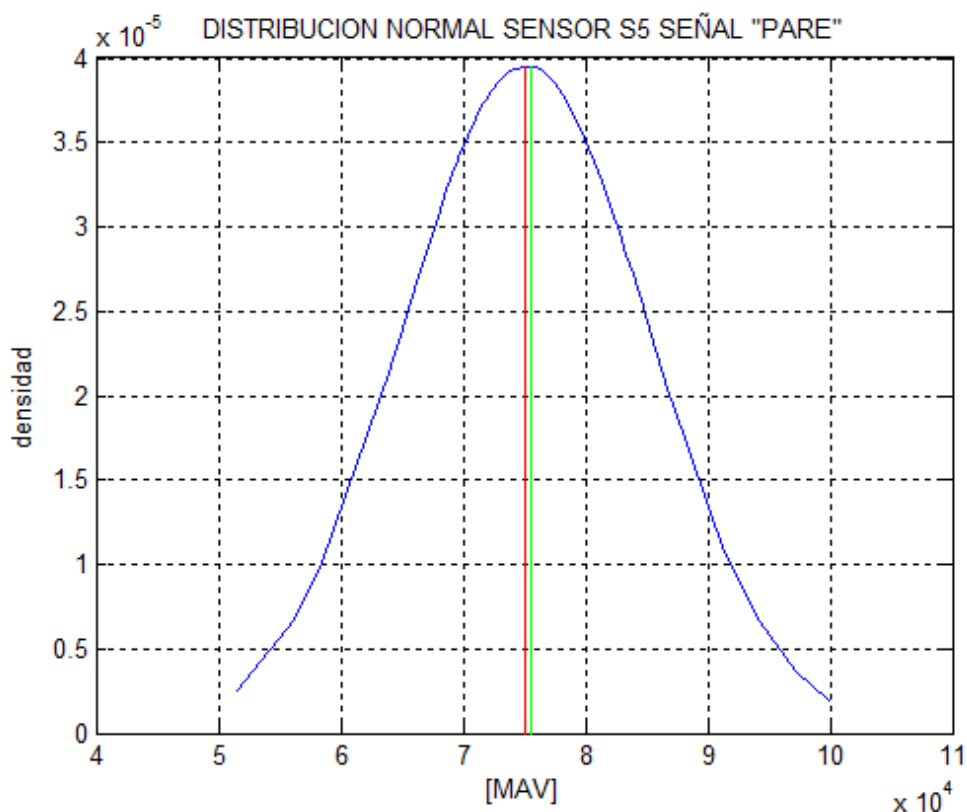


Figura 2.5: Gráfica de distribución normal sensor S5 señal "pare".

En la figura 2.5 se observa la gráfica de la distribución normal para el sensor S5 correspondiente a los datos obtenidos de la señal "pare", en color azul. La curva de color rojo corresponde a la media, mientras que la curva de color verde corresponde a la mediana. El valor de la media es aproximadamente igual al valor de la mediana, por esta razón, su gráfica se presenta de forma simétrica.

De acuerdo a lo anteriormente examinado, el análisis de los datos de los sensores que componen la MYO, serán contrastados con los valores de las medianas

respectivas de cada sensor, ya que, para este caso, la mediana es un dato más robusto entorno a los valores extremos presentados en los datos analizados.

2.3.2 Curtosis

Para el análisis de la curtosis, se ha considerado los cuatro valores cercanos a cero, ya que de esta manera se puede asegurar que los datos poseen una distribución mesocúrtica.

Sensor	Mediana [MAV]	Curtosis
S1	3093,50	43,65181
S2	4491,50	30,345488
S3	6627,50	24,245607
S4	8682,50	12,954554
S5	12664,00	0,9525285
S6	5592,50	1,3388131
S7	4820,75	1,7755117
S8	2930,00	0,7769212

Tabla 4: Valores de curtosis para señal de “siga”.

En la tabla 4 se presentan los valores de curtosis obtenidos de todos los sensores para la señal de “siga”.

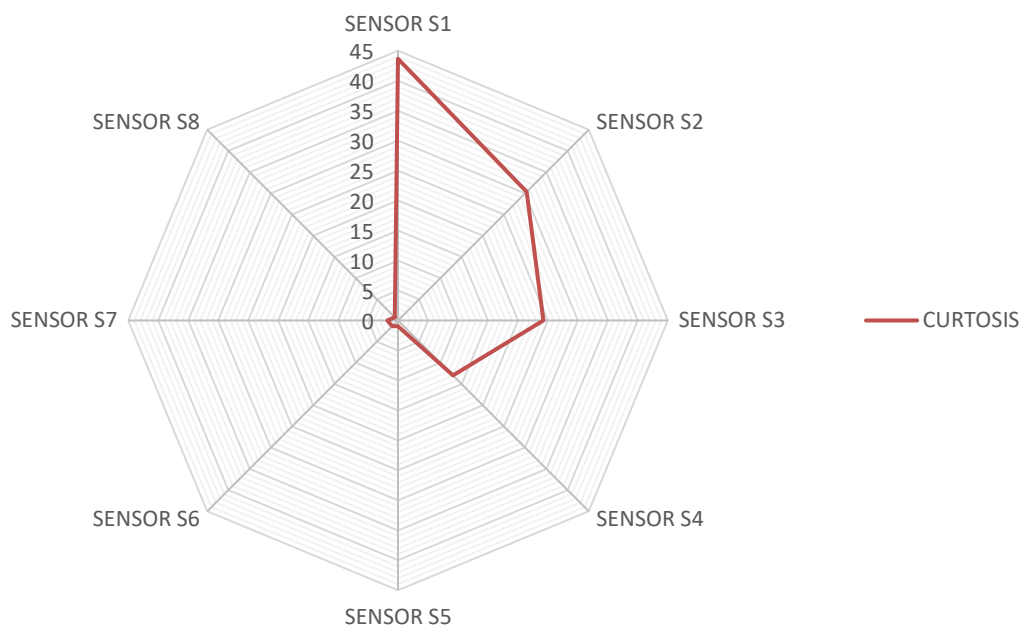


Figura 2.6: Análisis de curtosis para señal de “siga”.

En la figura 2.6 se observan los distintos valores de curtosis obtenidos de cada uno de los sensores para la señal de “siga”. Los sensores S5, S6, S7 y S8 presentan valores muy próximos a cero, lo que indica que los datos obtenidos de los sensores en mención, poseen un apuntamiento normal o mesocúrtica; es decir, como el de una distribución normal.

Sensor	Mediana [MAV]	Curtosis
S1	4855,25	1,4513492
S2	5620	1,5498374
S3	10817,25	2,502234
S4	35819,5	1,4029408
S5	75590	0,2054132
S6	34874,25	0,9416855
S7	36010,25	0,4070262
S8	6981,25	5,7610726

Tabla 5: Valores de curtosis para señal de “pare”.

En la tabla 5 se presentan los valores de curtosis obtenidos de todos los sensores para la señal de “pare”.

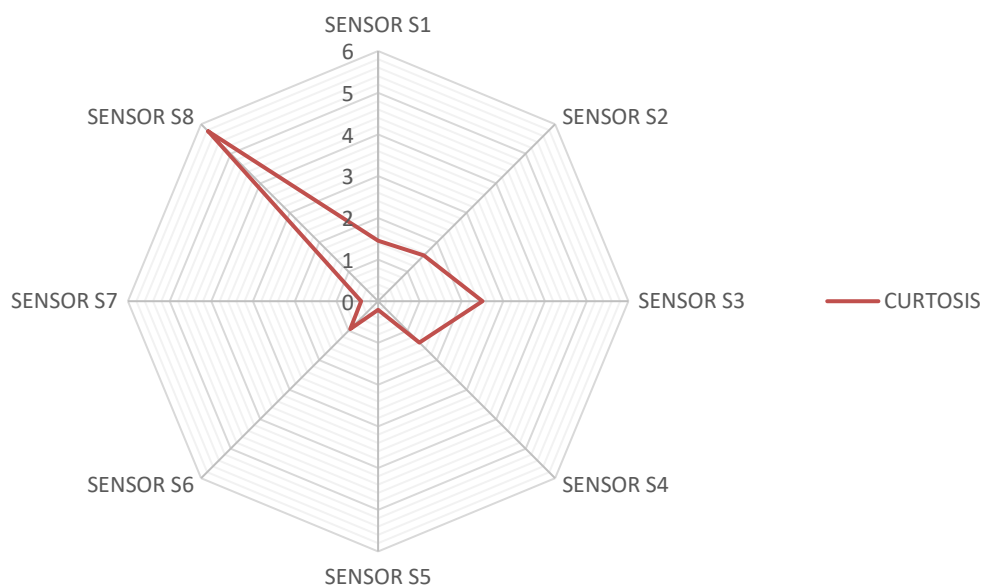


Figura 2.7: Análisis de curtosis para señal de “pare”.

En la figura 2.7 se observan los distintos valores de curtosis obtenidos de cada uno

de los sensores para la señal de “siga”. Los sensores S4, S5, S6 y S7 presentan valores muy próximos a cero, lo que indica que los datos obtenidos de los sensores en mención, poseen un apuntamiento normal o mesocúrtica; es decir, como el de una distribución normal.

Lo observado y analizado en las figuras 2.6 y 2.7, indica que la curtosis obtenida para los sensores indicados se aproxima a una distribución normal o mesocúrtica, debido a que sus valores son cercanos a cero. Este es el comportamiento que se busca obtener para la efectiva selección de los sensores que serán analizados posteriormente.

2.3.3 Límites mínimos y máximos que definen la validez de un dato

Para el análisis de los límites que definan la validez de los datos correspondientes a los sensores que componen la MYO, se ha considerado el rango comprendido por el mínimo y máximo de cada muestra; además de un análisis de los cuartiles correspondientes.

Sensor	Q1 [MAV]	Q3 [MAV]	Rango entre cuartiles [MAV]	Min [MAV]	Max [MAV]	Rango [MAV]
S1	2861,13	3831,88	970,75	2274,00	19826,50	17552,50
S2	4102,25	5335,63	1233,38	2740,00	25048,00	22308,00
S3	6120,50	7978,50	1858,00	3232,50	24426,50	21194,00
S4	7869,75	9782,63	1912,88	6042,00	19380,00	13338,00
S5	12154,63	13577,50	1422,88	10554,50	16511,00	5956,50
S6	4522,75	7764,63	3241,88	3650,50	14568,00	10917,50
S7	4118,13	5716,00	1597,88	2533,00	9603,50	7070,50
S8	2491,00	3384,75	893,75	1990,50	5807,00	3816,50

Tabla 6: Valores de rango de datos para señal de “siga”.

En la tabla 6 se observan los valores obtenidos para los cuartiles, el rango entre cuartiles, valores mínimos y máximos de datos y su rango respectivo de cada uno de los sensores para la señal de “siga”.

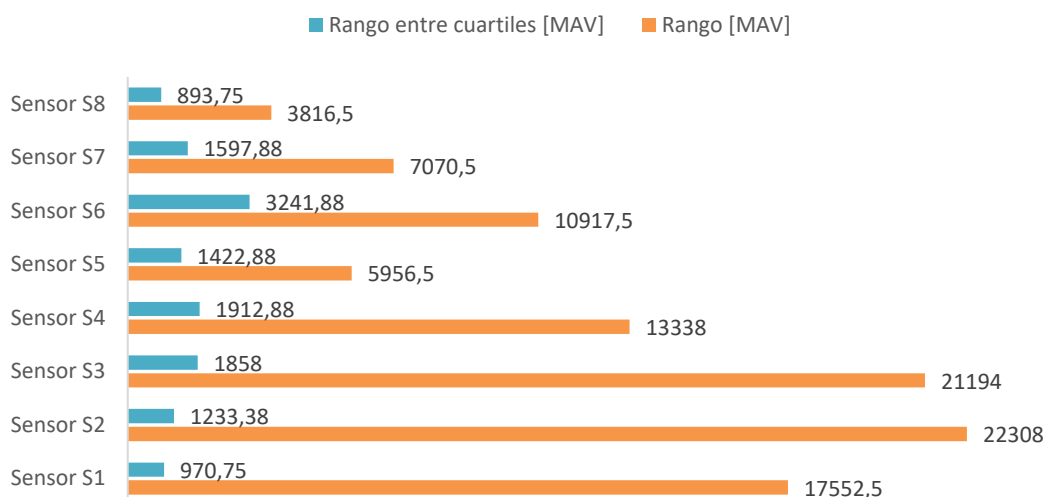


Figura 2.8: Análisis de rango de datos para señal de “siga”.

En la figura 2.8 se observan los valores obtenidos tanto para el rango entre cuartiles (color celeste) como para el rango de datos (color naranja) de cada uno de los sensores para la señal de “siga”. Se puede observar que el rango entre cuartiles es mucho menor que el rango generado por los valores máximos y mínimos existentes para cada uno de los sensores.

Sensor	Q1 [MAV]	Q3 [MAV]	Rango entre cuartiles [MAV]	Min [MAV]	Max [MAV]	Rango [MAV]
S1	4245,00	5552,75	1307,75	3319,50	8482,00	5162,50
S2	4932,13	6812,38	1880,25	3550,50	11782,50	8232,00
S3	9371,38	12619,25	3247,88	5819,50	22799,00	16979,50
S4	31494,63	40773,25	9278,63	21952,00	64714,50	42762,50
S5	69525,63	81551,75	12026,13	51525,50	99841,00	48315,50
S6	27108,75	44553,38	17444,63	16297,50	72431,50	56134,00
S7	31297,00	42220,63	10923,63	18752,00	60228,50	41476,50
S8	6526,38	7916,38	1390,00	4824,00	16055,00	11231,00

Tabla 7: Valores de rango de datos para señal de “pare”.

En la tabla 7 se observan los valores obtenidos para los cuartiles, el rango entre cuartiles, valores mínimos y máximos de datos y su rango respectivo de cada uno de los sensores para la señal de “pare”.

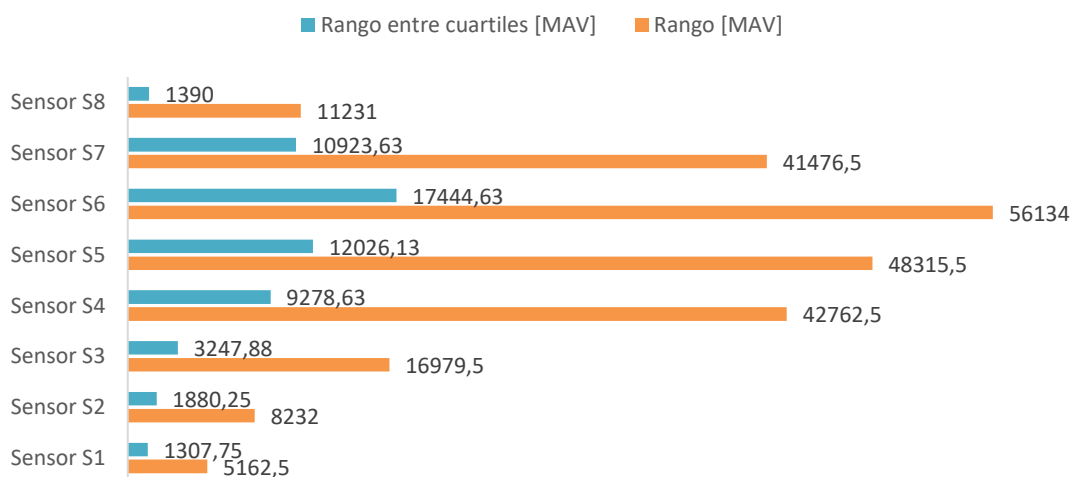


Figura 2.9: Análisis de rango de datos para señal de “pare”.

En la figura 2.9 se observan los valores obtenidos tanto para el rango entre cuartiles (color celeste) como para el rango de datos (color naranja) de cada uno de los sensores para la señal de “pare”. Se puede observar que el rango entre cuartiles es mucho menor que el rango generado por los valores máximos y mínimos existentes para cada uno de los sensores.

De acuerdo al tipo de medida de tendencia central, en nuestro caso la mediana, la medida de dispersión adecuada puede ser tanto el rango entre valores mínimo y máximo como el rango entre cuartiles. La condicionante para seleccionar el uno o el otro, depende totalmente de los valores considerados como valores extremos de los datos, ya que el rango entre valores máximo y mínimo es muy sensible a la variación de valores extremos, mientras que el rango entre cuartiles es una medida robusta a dichos valores extremos.

2.4 Selección y validación de sensores

De acuerdo a las observaciones realizadas previamente, se ha considerado como medida de tendencia central, los valores de las medianas, debido a que todas las distribuciones tienen forma asimétrica y sesgada, lo que indica que la mediana es una medida robusta para este tipo de distribuciones.

Conforme al valor obtenido de curtosis, se han seleccionado los sensores S4, S5, S6 y S7 para la señal de “pare”, mientras que para la señal de “siga”, se han seleccionado los sensores S5, S6, S7 y S8. Estos grupos de sensores fueron escogidos debido a que presentan su valor de curtosis más cercano al cero, lo que indica que se su nivel de apuntamiento se aproxima al de una distribución normal.

Ahora, se toman en consideración los valores de las medianas, cuartiles y rango de cuartiles obtenidos para los sensores S4, S5, S6, S7 y S8, observando que no existan valores comunes entre ellos.

Señal "siga"				Señal "pare"		
Sensor	Mediana	Q1	Q3	Mediana	Q1	Q3
S4	8682,50	7869,75	9782,63	35819,50	31494,63	40773,25
S5	12664,00	12154,63	13577,50	75590,00	69525,63	81551,75
S6	5592,50	4522,75	7764,63	34874,25	27108,75	44553,38
S7	4820,75	4118,13	5716,00	36010,25	31297,00	42220,63
S8	2930,00	2491,00	3384,75	6981,25	6526,38	7916,38

Tabla 8: Valores de medianas y cuartiles entre señales de "siga" y "pare" para sensores seleccionados.

En la tabla 8 se observan los sensores seleccionados para el estudio de la red neuronal, con los valores de medianas y cuartiles correspondientes para las señales de "siga" y "pare".

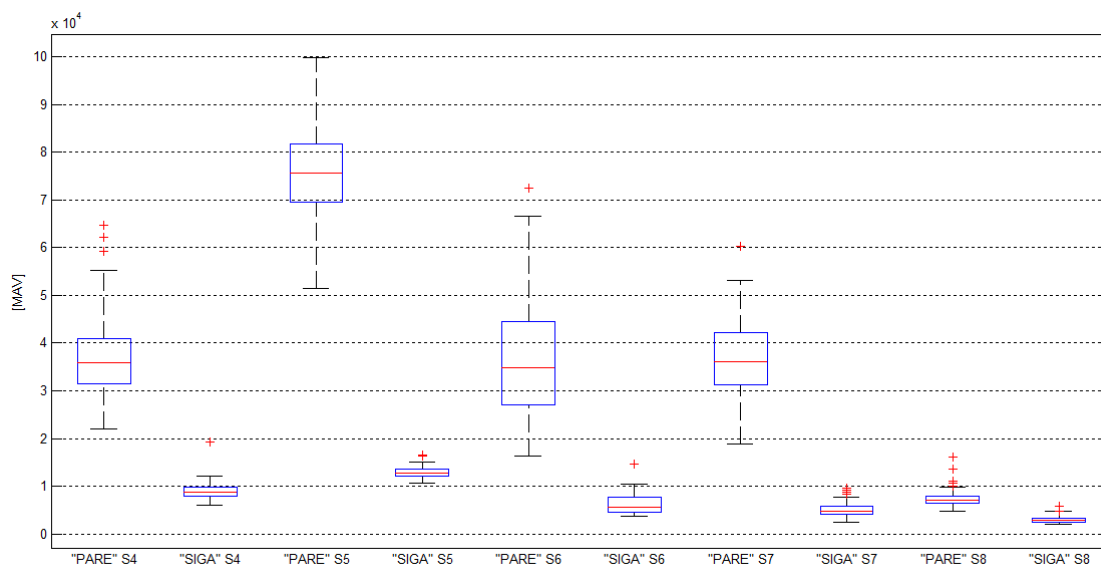


Figura 2.10: Diagrama de caja de señales "pare" y "siga" correspondiente a los sensores S4, S5, S6, S7 y S8.

En la figura 2.10 se pueden apreciar los diagramas de caja para cada uno de los sensores S4, S5, S6, S7 y S8; en el cual se muestra la relación existente entre los

valores correspondientes a la señal de “siga” y los valores correspondientes a la señal de “pare”.

Los diagramas de caja muestran un recuadro o “caja” que se compone por el tercer cuartil en su arista superior, el primer cuartil en su arista inferior, la línea media que divide la caja corresponde al valor de la mediana, los “bigotes” tanto en la parte superior como en la inferior indican los límites permitidos, y las pequeñas marcas en forma de cruz que se encuentran más allá de los “bigotes”, corresponden a datos extraños o aberrantes.

Se observa, por cada sensor escogido, que ningún valor de dato correspondiente a la señal “siga” se intercepta con valores de datos correspondiente a la señal de “pare”; lo que significa que, estadísticamente, durante la ejecución de los movimientos respectivos, el controlador se asegura de la correcta identificación del mismo. Esto se acoteja como un complemento al funcionamiento de la red neuronal.

Por lo anteriormente expuesto, los sensores S4, S5, S6, S7 y S8 son considerados para formar parte del posterior estudio de la red neuronal.

2.5 Análisis de la red neuronal obtenida

Para el reconocimiento de patrones de la red neuronal, se plantea la matriz de entrenamiento compuesta de muestras tanto de señales de “siga” y de señales de “pare”. Cabe destacar que todos estos datos han sido previamente tratados mediante las herramientas estadísticas explicadas en secciones previas.

Para el uso de la red neuronal, es necesario garantizar que el factor de correlación de los datos sea cercano a 0 entre los sensores de las muestras de distintas señales. De esta manera se avala de que el dato es óptimo para su uso dentro del aprendizaje y pruebas de la red neuronal.

		Señal "pare"				
		S4	S5	S6	S7	S8
Señal "siga"	S4	0,01	-	-	-	-
	S5	-	0,14	-	-	-
	S6	-	-	-0,43	-	-
	S7	-	-	-	0,14	-
	S8	-	-	-	-	-0,21

Tabla 9: Análisis de correlación de sensores entre señales de la base de datos existente.

En la tabla 9 se observan los distintos valores de correlación existentes entre los sensores y las señales tanto de "siga" como la de "pare". Todos los valores presentan una tendencia hacia el cero, lo que indica que los puntos son muy dispersos y su relación lineal es de débil a nula. De esta manera se concluye que las muestras registradas en la base de datos son aptas para ingresar a la red neuronal.

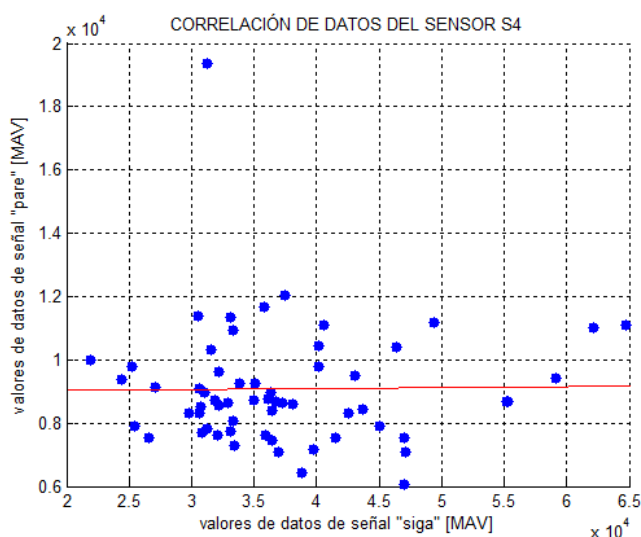


Figura 2.11(a): Correlación existente entre los datos del sensor S4.

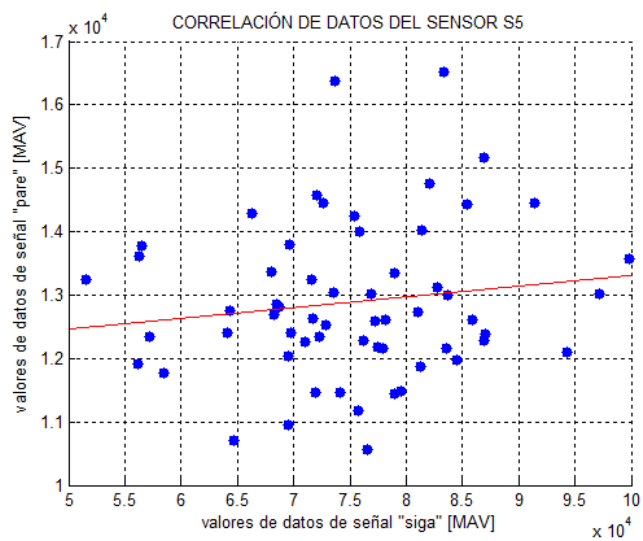


Figura 2.11(b): Correlación existente entre los datos del sensor S5.

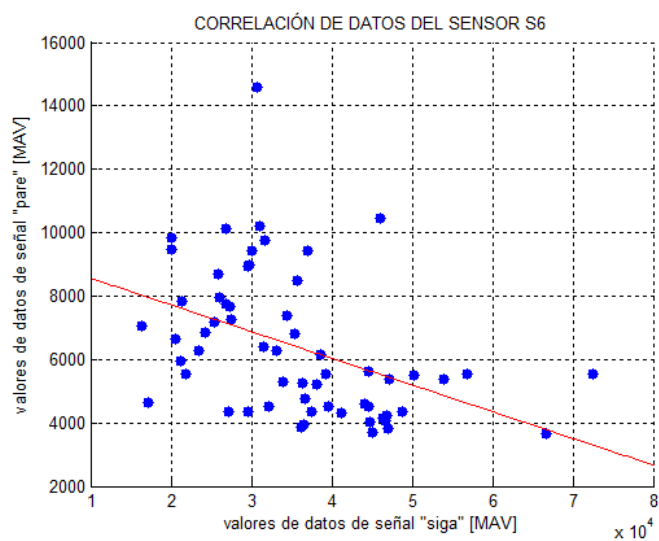


Figura 2.11(c): Correlación existente entre los datos del sensor S6.

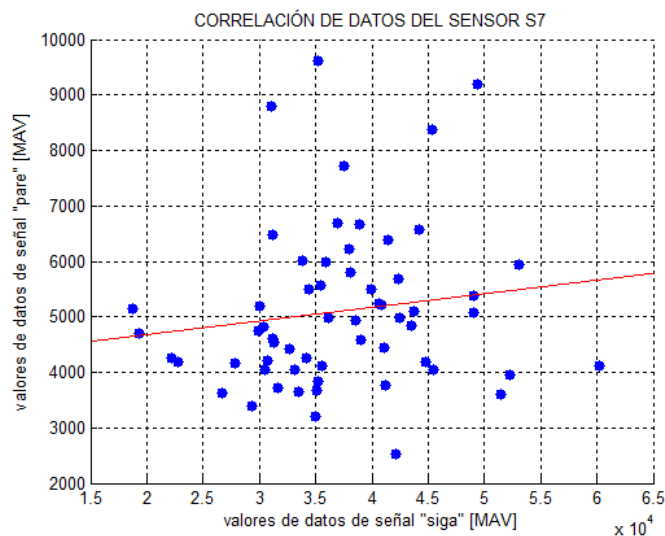


Figura 2.11(d): Correlación existente entre los datos del sensor S7.

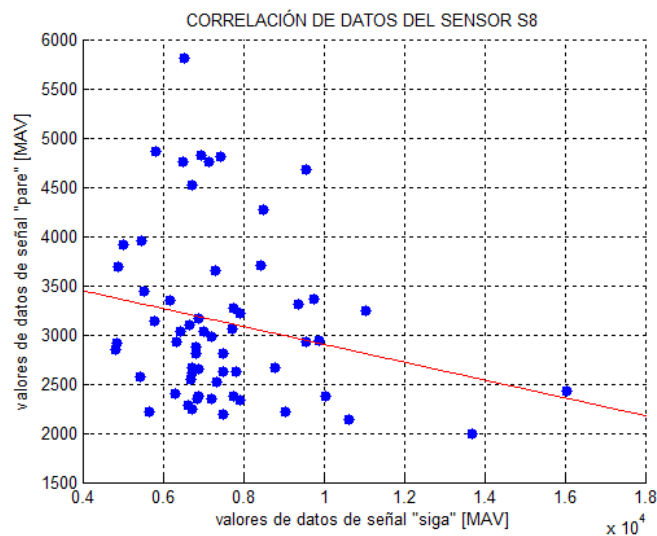


Figura 2.11(e): Correlación existente entre los datos del sensor S8.

La figura 2.11 muestra las gráficas de dispersión de datos de los sensores S4 (figura 2.11(a)), S5 (figura 2.11(b)), S6 (figura 2.11(c)), S7 (figura 2.11(d)) y S8 (figura 2.11(e)) para la información obtenida en la señal de "sig" versus la señal de "pare". La curva de color rojo indica la correlación existente entre los datos.

Comprobando con los valores obtenidos en la tabla 9, se observa en la figura 2.11(a) que el sensor S4 posee un coeficiente de correlación de aproximadamente 0; en la figura 2.11(b) y figura 2.11(d) se observa que tanto el sensor S5 y el sensor S7 respectivamente, poseen un coeficiente de correlación menos cercano a cero y de valor positivo, lo que se refleja en la pendiente positiva presente en la curva de color rojo. De la misma manera que el caso previo, no existe relación entre los datos de “siga” con los datos de “pare”; en la figura 2.11(c) y figura 2.11(e) se observa que tanto el sensor S6 y el sensor S8 respectivamente, poseen un coeficiente de correlación menos cercano a cero y de valor negativo, lo que se refleja en la pendiente negativa presente en la curva de color rojo. En todas las gráficas mencionadas previamente, no existe relación entre los datos de “siga” con los datos de “pare”

Al no existir un valor de correlación lo más cercano a 1 o -1, se concluye que los datos obtenidos de los sensores tanto para la señal de “pare” y “siga” no se verán afectados al momento de analizar el tipo de señal emitida, lo que comprende una gran ventaja al momento de definir las matrices de entrenamiento, validación y pruebas para la generación de la red neuronal.

Luego de establecida la definición de las matrices de entrenamiento, validación y pruebas, se plantea la codificación para las salidas esperadas de la red. En este caso, deberá ser 2 valores o códigos asignados a cada señal.

Siga	Pare
1	0
0	1

Tabla 10: Codificación de la red neuronal.

La tabla 10 muestra la codificación utilizada dentro de la red neuronal para la identificación de las señales de “siga” y “pare”.

Mediante el uso de la herramienta *nntool* dentro del entorno de Matlab, se procede con la creación de la red neuronal de tipo retro propagación, o también llamada

propagación hacia atrás de errores, el cual es un método de cálculo de gradiente usado en algoritmos de aprendizaje supervisado. La importancia de este proceso consiste en que, a medida que se va entrenando la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que todas las neuronas aprenden a reconocer distintas características del espacio total de la entrada. Luego de su entrenamiento, al presentarse datos de entrada que contengan ruido o sean datos incompletos, las neuronas de la capa oculta responderán con una salida activa si el nuevo paquete de datos de entrada posee un patrón semejante a la característica que las neuronas individuales hayan aprendido a reconocer durante el entrenamiento.

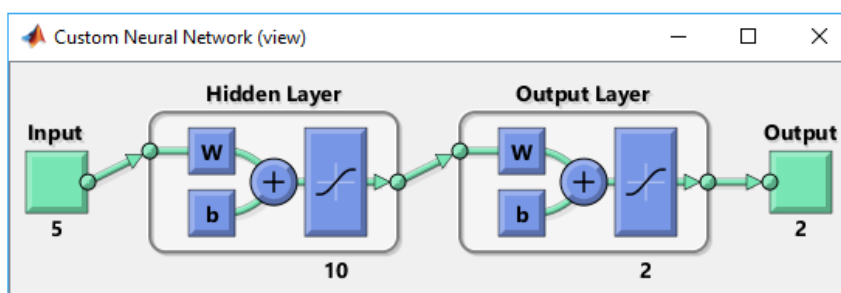


Figura 2.12: Parámetros de la red neuronal.

En la figura 2.12 se observan los parámetros de la red neuronal: el bloque correspondiente a los datos de entrada (correspondiente al área bajo la curva de los datos de los sensores S4, S5, S6, S7 y S8 respectivamente), la capa oculta compuesta por 10 neuronas y la capa de salida compuesta por 2 neuronas, la cual se encarga de enviar los valores establecidos en la tabla 10 correspondiente a los valores de codificación de la red neuronal.

Debido a que no son necesarios los valores esperados de un resultado anterior, no existe la necesidad de generar más capas ocultas, ya que los datos son procesados al mismo tiempo generando una única decisión, característica necesaria que cumple la única capa oculta existente dentro de la red neuronal.

Luego de generada la red neuronal con las características mencionadas en los párrafos previos, se procede con el análisis de su desempeño, gradiente y correlación existente.

La necesidad de la cantidad de entrenamientos que se le deba dar a la red neuronal dependerá de que su valor de desempeño tienda a cero, de igual manera para los valores obtenidos de gradiente, y la correlación de datos existentes debe aproximarse a 1. Con estas tres características, se puede asegurar un perfecto desempeño de la red neuronal.

Cabe destacar que, para verificar la efectividad obtenida de la red neuronal, los vectores o datos de entrada deberán corresponder únicamente a las señales de estudio, es decir, la señal de “pare” y la señal de “siga”.

CAPÍTULO 3

3. RESULTADOS

3.1 Resultados obtenidos de la red neuronal aplicada

Luego de haber definido todos los parámetros que conforman la red neuronal como las matrices de entrenamiento y la matriz de resultados esperados, se procede con el entrenamiento de la misma a partir del esquema de una red neuronal con las siguientes características:

- Tipo de red neuronal: Retro propagación.
- Tipo de entrenamiento: gradiente descendiente.
- Función de aprendizaje: Error cuadrático medio.
- Función de activación: Sigmoide.
- Número de neuronas en la capa oculta: 10.

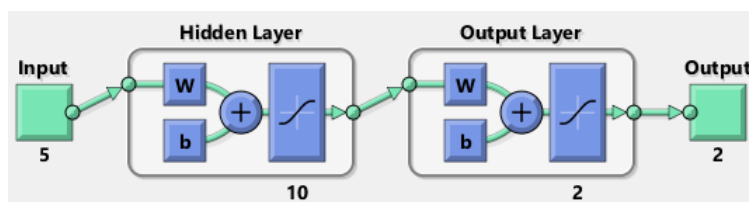


Figura 3.1: Diagrama de la red neuronal aplicada.

En la figura 3.1 se muestra el diagrama de la red neuronal aplicada, compuesta por los diagramas de bloques de entrada, capa oculta, capa de salida y salida.

3.1.1 Resultados obtenidos de la red neuronal aplicada al primer entrenamiento

Los parámetros aplicados a la red neuronal para el primer entrenamiento de la misma, son los siguientes:

- Épocas: 1000
- Iteraciones de épocas: 1000

- Error máximo: 1.5
- Gradiente mínimo: 1×10^{-5}

Durante el primer entrenamiento de la red neuronal bajo los parámetros previamente establecidos, se obtienen los siguientes resultados:

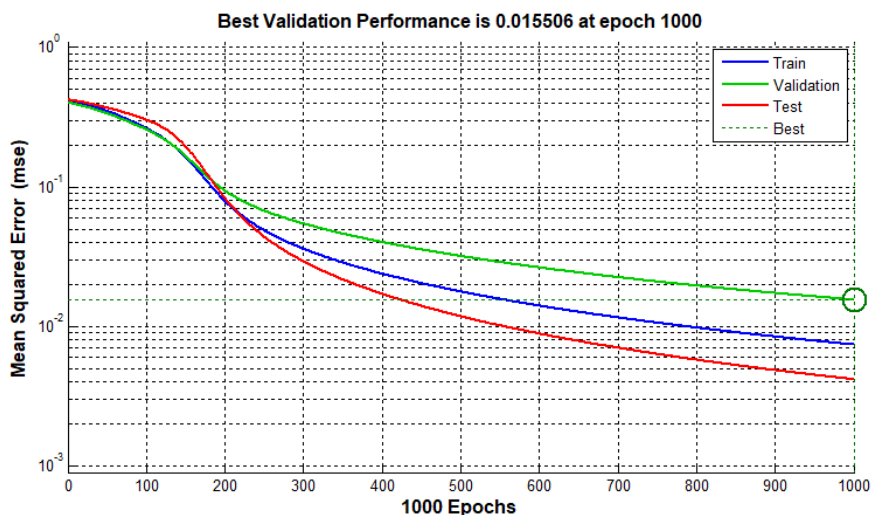


Figura 4.2(a): Validación del desempeño para la red neuronal luego de su primer entrenamiento.

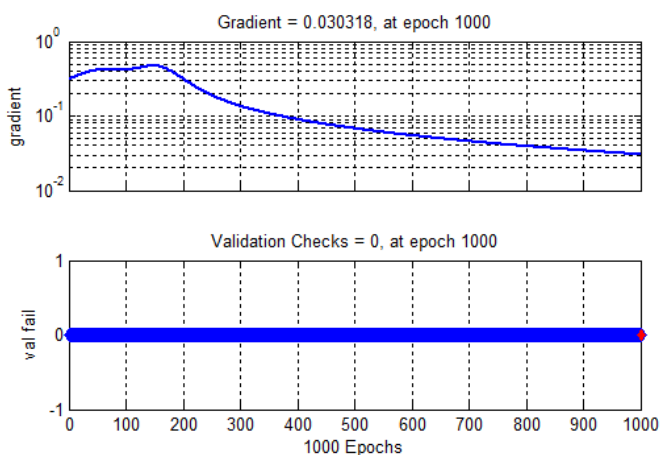


Figura 5.2(b): Gradiente obtenido por la red neuronal luego de su primer entrenamiento.

En la figura 3.2(a) se observa que la validación del desempeño de la red neuronal aplicada presenta un valor de error del 1.55%, mientras que en la figura 3.2(b) se observa que el valor del gradiente obtenido es de 0.030318.

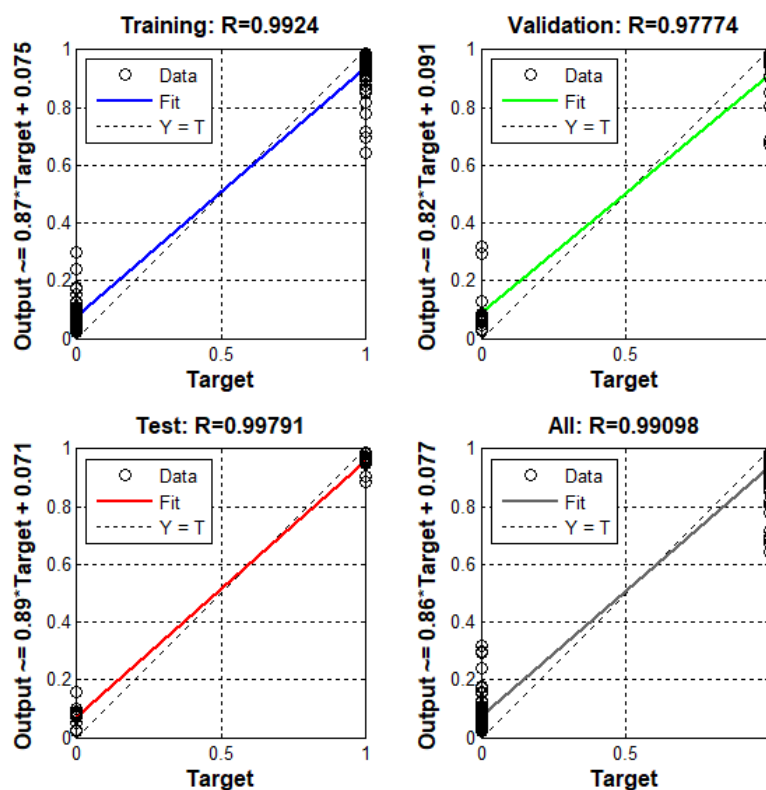


Figura 6.3: Correlación existente entre los datos de entrada y salida de la red neuronal aplicada luego de su primer entrenamiento.

La figura 3.3 presenta la gráfica de la relación existente entre los datos de entrada y los datos de salida esperados por la red neuronal, es decir, la correlación existente entre los valores de la matriz de entrenamiento y la matriz de valores esperados es de 0.99, valor que puede considerarse aceptable debido a su cercanía a 1, lo cual garantizará un error mínimo entre los valores de salida esperados.

Luego de obtenida la red neuronal, se procede con las pruebas de la misma por medio de un banco de datos de prueba constituido por 5 datos de señal "siga" y 5 datos de señal "pare".

		DATO DE PRUEBA "pare"					PROMEDIO
		DATO 01	DATO 02	DATO 03	DATO 04	DATO 05	
RESPUESTA DE LA RED	"pare"	76,85%	67,38%	66,69%	74,49%	22,82%	61,65%
	"siga"	19,28%	41,55%	37,00%	66,58%	43,18%	41,52%

Tabla 11(a): Tabla de respuestas de la red neuronal luego de su primer entrenamiento para datos de prueba de la señal "pare".

		DATO DE PRUEBA "siga"					PROMEDIO
		DATO 01	DATO 02	DATO 03	DATO 04	DATO 05	
RESPUESTA DE LA RED	"pare"	6,72%	6,05%	5,99%	6,38%	6,11%	6,25%
	"siga"	84,98%	95,76%	94,31%	96,80%	95,45%	93,46%

Tabla 11(b): Tabla de respuestas de la red neuronal luego de su primer entrenamiento para datos de prueba de la señal "siga".

La tabla 11(a) presenta los valores de respuesta obtenidos de la red neuronal mediante un banco de datos de prueba correspondientes a la señal "pare", la tabla 11(b) presenta los valores de respuesta obtenidos de la red neuronal mediante un banco de datos de prueba correspondientes a la señal "siga".

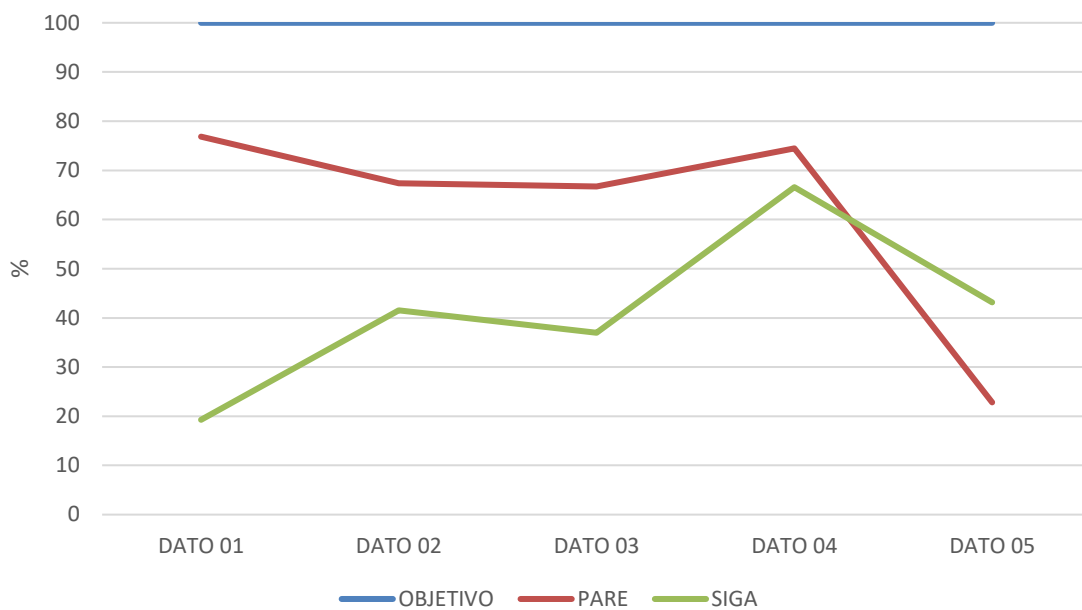


Figura 7.4(a): Respuesta de la red neuronal luego de su primer entrenamiento para datos de prueba de la señal “pare”.

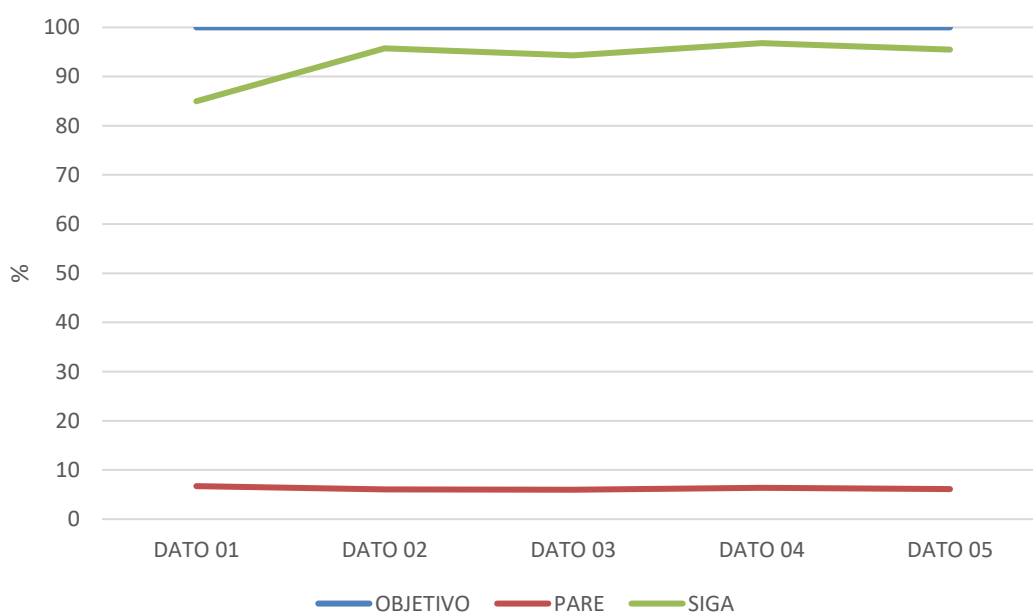


Figura 8.4(b): Respuesta de la red neuronal luego de su primer entrenamiento para datos de prueba de la señal “siga”.

En la figura 3.4(a) se observa la respuesta de la red neuronal con datos de prueba de señal “pare” luego de su primer entrenamiento. El objetivo se indica mediante la curva de color azul, donde se espera obtener un valor muy próximo al 100%; la curva de color rojo representa la respuesta obtenida como señal “pare”, mientras que la curva de color verde representa la respuesta obtenida como señal “siga”. Se evidencia que la curva de color rojo sobrepasa la curva de color verde, acercándose hacia la curva azul, el objetivo; sin embargo, existen valores donde la curva de color verde tiene mayor representación que la curva de color rojo.

En la figura 3.4(b) se observa la respuesta de la red neuronal con datos de prueba de señal “siga” luego de su primer entrenamiento. El objetivo se indica mediante la curva de color azul, donde se espera obtener un valor muy próximo al 100%; la curva de color rojo representa la respuesta obtenida como señal “pare”, mientras que la curva de color verde representa la respuesta obtenida como señal “siga”. Se evidencia que la curva de color verde se encuentra muy próxima a la curva objetivo, mientras que la curva de color rojo se encuentra muy cercana al cero.

De acuerdo a lo analizado en las figuras 3.4(a) y 3.4(b), es necesario realizar un segundo entrenamiento a la red neuronal, debido a que, a pesar de que en las pruebas con señal “siga”, su respuesta fue satisfactoria con un porcentaje promedio del 93.46% de efectividad, las pruebas con la señal “pare” no fueron así, presentando un porcentaje promedio del 61.65% de efectividad.

3.1.2 Resultados obtenidos de la red neuronal aplicada al segundo entrenamiento

Los parámetros aplicados a la red neuronal para el segundo entrenamiento de la misma, son los siguientes:

- Épocas: 10000
- Iteraciones de épocas: 10000
- Error máximo: 0.06
- Gradiente mínimo: 1×10^{-5}

Durante el primer entrenamiento de la red neuronal bajo los parámetros previamente

establecidos, se obtienen los siguientes resultados:

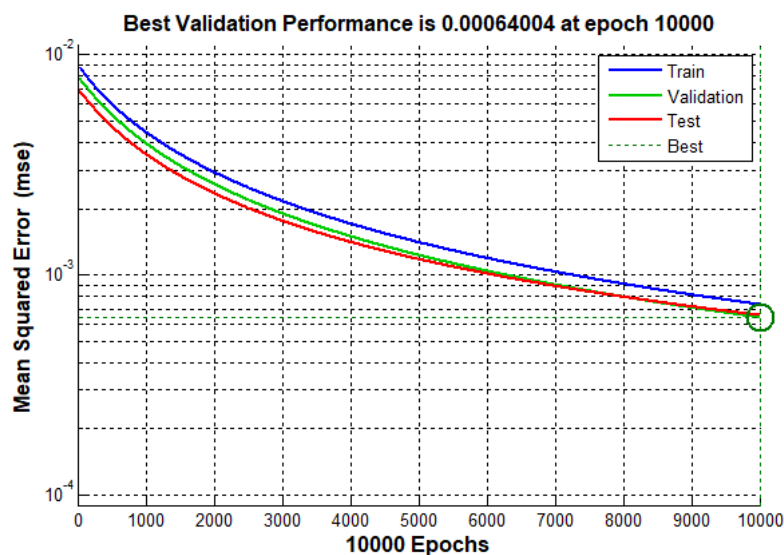


Figura 9.5(a): Validación del desempeño para la red neuronal luego de su segundo entrenamiento.

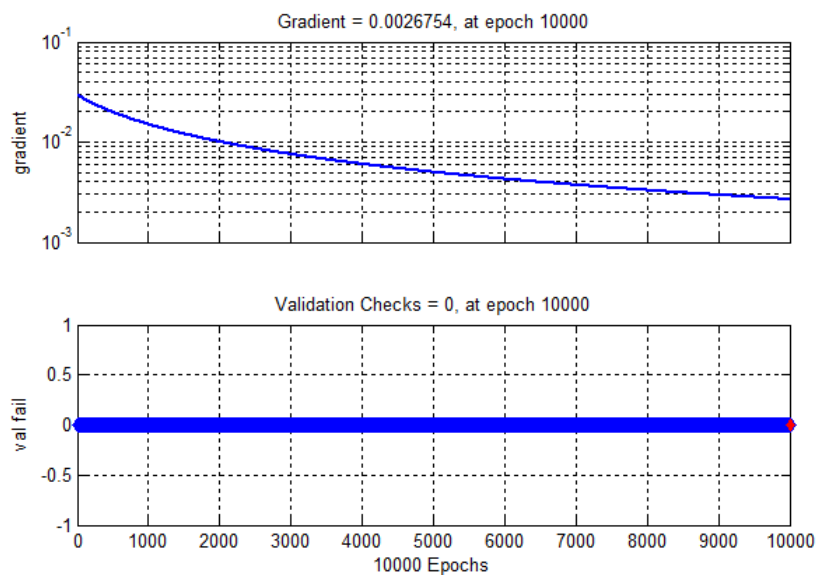


Figura 10.5(b): Gradiente obtenido por la red neuronal luego de su segundo entrenamiento.

En la figura 3.5(a) se observa que la validación del desempeño de la red neuronal aplicada presenta un valor de error del 0.064%, mientras que en la figura 3.5(b) se observa que el valor del gradiente obtenido es de 0.0026754.

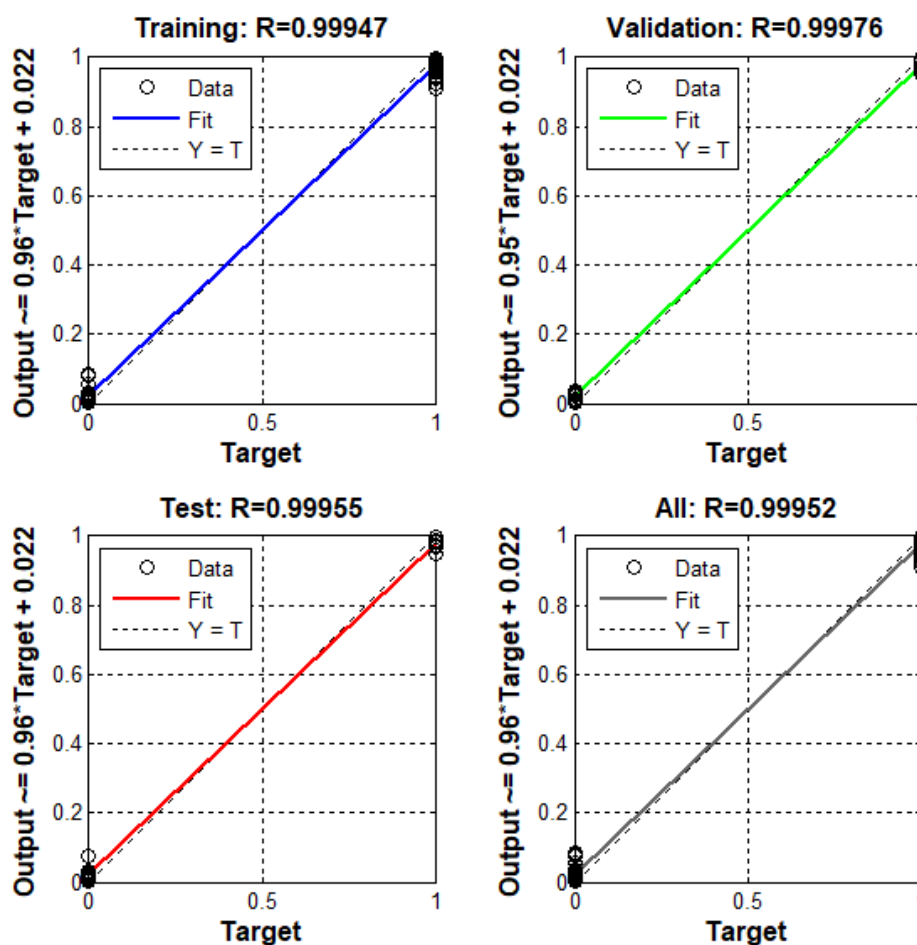


Figura 11.6: Correlación existente entre los datos de entrada y salida de la red neuronal aplicada luego de su segundo entrenamiento.

La figura 3.6 presenta la gráfica de la relación existente entre los datos de entrada y los datos de salida esperados por la red neuronal, es decir, la correlación existente entre los valores de la matriz de entrenamiento y la matriz de valores esperados es de 0.9995, valor que puede considerarse aceptable debido a su cercanía a 1, lo cual garantizará un error mínimo entre los valores de salida esperados.

Luego de obtenido el segundo entrenamiento de la red neuronal, se procede con las pruebas de la misma por medio de un banco de datos de prueba constituido por 5 datos de señal "siga" y 5 datos de señal "pare".

		DATO DE PRUEBA "pare"					PROMEDIO
		DATO 01	DATO 02	DATO 03	DATO 04	DATO 05	
RESPUESTA DE LA RED	"pare"	99,03%	99,76%	98,94%	98,67%	94,19%	98,12%
	"siga"	3,06%	13,39%	3,51%	2,72%	3,72%	5,28%

Tabla 12(a): Tabla de respuestas de la red neuronal luego de su segundo entrenamiento para datos de prueba de la señal "pare".

		DATO DE PRUEBA "siga"					PROMEDIO
		DATO 01	DATO 02	DATO 03	DATO 04	DATO 05	
RESPUESTA DE LA RED	"pare"	3,12%	2,24%	2,35%	2,19%	2,31%	2,44%
	"siga"	97,27%	97,58%	97,54%	97,63%	97,58%	97,52%

Tabla 12(b): Tabla de respuestas de la red neuronal luego de su segundo entrenamiento para datos de prueba de la señal "siga".

La tabla 12(a) presenta los valores de respuesta obtenidos de la red neuronal luego de su segundo entrenamiento mediante un banco de datos de prueba correspondientes a la señal "pare", la tabla 12(b) presenta los valores de respuesta obtenidos de la red neuronal luego de su segundo entrenamiento mediante un banco de datos de prueba correspondientes a la señal "siga".

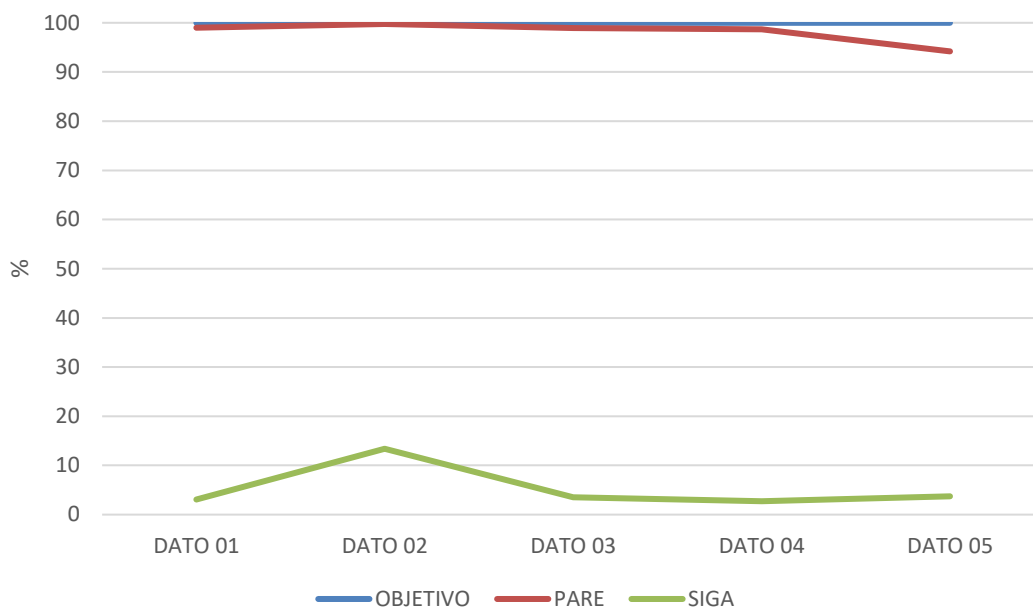


Figura 12.7(a): Respuesta de la red neuronal luego de su segundo entrenamiento para datos de prueba de la señal “pare”.

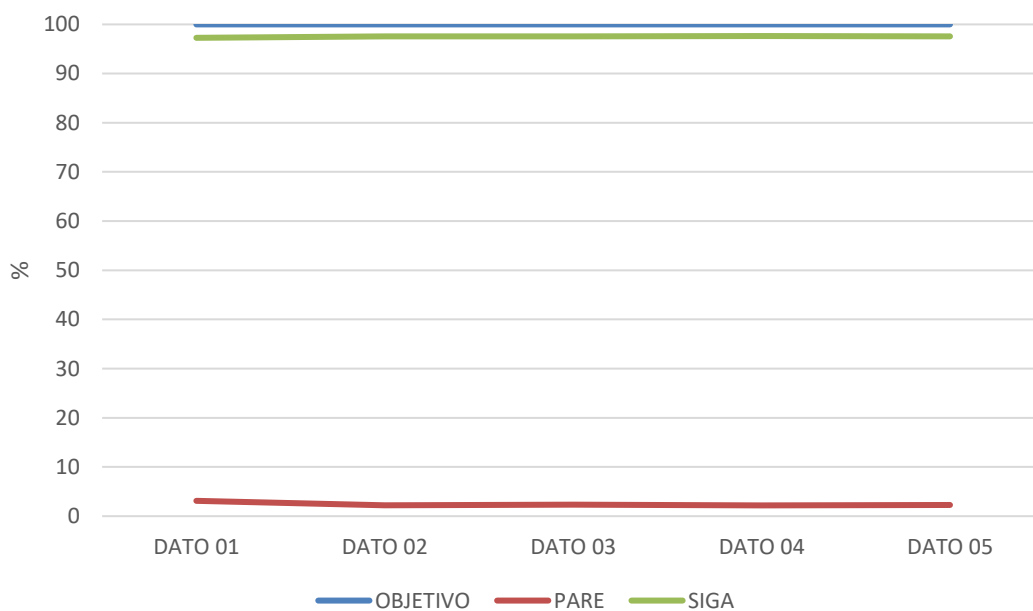


Figura 13.7(b): Respuesta de la red neuronal luego de su segundo entrenamiento para datos de prueba de la señal “siga”.

En la figura 3.7(a) se observa la respuesta de la red neuronal con datos de prueba de señal “pare” luego de su segundo entrenamiento. Se evidencia una mejora en su respuesta comparada con su primer entrenamiento, ya que la curva de color rojo se encuentra muy próxima a la curva objetivo, mientras que la curva de color verde está mucho más próxima al cero.

En la figura 3.7(b) se observa la respuesta de la red neuronal con datos de prueba de señal “siga” luego de su segundo entrenamiento. Se evidencia que la curva de color verde se encuentra más próxima a la curva objetivo, mientras que la curva de color rojo se encuentra muy cercana al cero.

De acuerdo a lo analizado en las figuras 3.7(a) y 3.7(b), mediante los ensayos realizados con los datos de prueba de señal “pare” y señal “siga”, se obtuvo un porcentaje de aproximación para la señal “siga” de 97.52%, mientras que para la señal “pare”, se obtuvo un porcentaje de aproximación de 98.12%; por consiguiente, se concluye que la red neuronal, en su segundo entrenamiento, cumple lo solicitado.

CONCLUSIONES

- Para el tratamiento de señales electromiográficas (sEMG), es importante que los datos sean normalizados para que de esta manera se realice el cálculo del área bajo la curva y así obtener el valor característico esperado, para luego proceder con el análisis de cada uno de los sensores que componen el dispositivo MYO.
- Para el proceso de entrenamiento de una red neuronal, es importante capturar información en diferentes estados que tiene una persona con el movimiento que se desea aprender, esto permitirá tener un banco de datos amplios y disminuir la cantidad de errores que puede aparecer durante el análisis de reconocimiento de movimientos.
- Es importante que los datos de entrenamiento enviadas a una red neuronal, tengan la misma característica en tiempo, gesto y estado al realizar el movimiento, ya que puede influir en el desarrollo de la interpretación del movimiento.
- La herramienta nntool de Matlab es primordial en la creación y entrenamiento de las redes neuronales. Para obtener los mejores resultados durante el entrenamiento, se debe tomar muy en cuenta el número de épocas y la cantidad de reentrenamientos que sean necesarios hasta que los valores obtenidos por la red neuronal, sean satisfactorios; es decir, su factor de correlación entre los datos de valores esperados y los datos de valores obtenidos sea más próximo a 1.
- La correlación de los datos es muy importante al momento de formar la matriz de entrenamiento de una red neuronal, considerando los sensores que contienen mayor grado de importancia mediante su análisis estadístico. Dicho factor deberá ser lo más cercano a 1 entre las muestras de una misma señal, y debajo de 0.5 entre las muestras de señales distintas, garantizando así un mejor desempeño en el desarrollo de la red

neuronal mediante el reconocimiento de patrones.

- La característica principal de la MYO usada dentro del desarrollo de este proyecto fue la de obtener la medición de las señales electromiográficas de los músculos que conforman el antebrazo; sin embargo, existen parámetros adicionales que podrán ser considerados al momento de mejorar la base de datos para el entrenamiento de la red neuronal, como por ejemplo, la señal del giroscopio, la cual puede complementar la información existente.

RECOMENDACIONES

- Se recomienda que los datos adquiridos de entrenamiento para la red neuronal sean en diferentes estados, días y haber realizado alguna actividad deportiva o en situaciones en la cual involucre un cansancio físico, ya que esto permitirá obtener una matriz de datos con información mucho más real.
- Para el envío de información remota a puntos lejanos para el procesamiento de los datos, es recomendable utilizar comunicación RFID ya que tiene un desplazamiento de cobertura de más de 100 metros y bajo costo.
- Es recomendable que, para el proceso de adquisición y recepción de datos, se utilice un dispositivo ordenador de placa reducida como son Arduino, Raspberry y Lattepanda, debido a que son dispositivos de bajo costo, pero potentes al momento de ejecutar algún desarrollo.
- Para simplificar el procesamiento de la información de la red neuronal, es recomendable que los valores correspondientes a la codificación de salidas esperadas sean entre unos y ceros; además de considerar no más de una capa oculta, debido a que el método de aprendizaje de la red no espera datos previos, es decir, todos los datos son procesados al mismo tiempo para obtener su valor objetivo, sin depender de condiciones previas.

BIBLIOGRAFÍA

- [1] M. Bouskela, M. Casseb, S. Bassi, C. De Luca, and M. Facchina, “La ruta hacia las smart cities: Migrando de una gestión tradicional a la ciudad inteligente,” 2016.
- [2] M. Ninacansaya and A. Ronald, “Análisis y diseño de un Sistema de Control de Trafico Vehicular Utilizando Semaforos Inteligentes con Tecnología Arduino,” 2016.
- [3] H. R. O. Aguirre, S. L. Salas, and V. M. L. Moreno, “SIMULACIÓN BASADA EN AGENTES PARA EL CONTROL INTELIGENTE DE SEMÁFOROS MEDIANTE LÓGICA DIFUSA,” *Pist. Educ.*, vol. 39, no. 128, 2018.
- [4] J. Camarena, L. Contreras, K. Moreno, M. Rodríguez, and C. Salazar, “Aplicaciones del IoT para el control de congestión vehicular,” in *Memorias de Congresos UTP*, 2018, vol. 1, no. 1, pp. 90–95.
- [5] G. Q. N. Patricio and O. G. J. Valeria, “Estudio De Una Red De Semáforos Inteligentes Para El Control Del Tránsito Vehicular De La Isla Trinitaria.,” Universidad De Guayaquil. Facultad De Ciencias Matemáticas Y Físicas. Carrera De Ingeniería En Networking Y Telecomunicaciones, 2017.
- [6] “Ecuador - Estadísticas de Transportes 2016 - Información general.” [Online]. Available: <https://anda.inec.gob.ec/anda/index.php/catalog/604>. [Accessed: 17-Jul-2019].
- [7] G. Cookson and B. Pishue, “INRIX Global Traffic Scorecard,” 2017.
- [8] E. Y. Arias, D. H. Albarracin, and N. F. Walteros, “Adquisición y tratamiento de señales mioeléctricas en extremidades superiores,” *Desarro. E INNOVACIÓN EN Ing.*, p. 492, 2017.
- [9] R. A. Blanco, A. M. C. Bernal, and M. P. Torres, “Diseño de un sistema difuso para el reconocimiento de la actividad muscular en señales de EMG superficiales/Design of a fuzzy logic system for muscular activity recognition

- using superficial EMG signals," *Int. J. Innov. Appl. Stud.*, vol. 19, no. 4, p. 729, 2017.
- [10] J. V. Ramírez, A. V. Lesso, and J. J. M. Nolasco, "EVALUACIÓN CUANTITATIVA DE LA ACTIVIDAD MIOELÉCTRICA," *JÓVENES EN LA Cienc.*, vol. 3, no. 2, pp. 2300–2305, 2017.
- [11] J. J. V. Mayor, R. M. Costa, A. Frizzera-Neto, and T. F. Bastos, "Decodificación de Movimientos Individuales de los Dedos y Agarre a Partir de Señales Mioeléctricas de Baja Densidad," *Rev. Iberoam. Automática e Informática Ind. RIAI*, vol. 14, no. 2, pp. 184–192, 2017.
- [12] E. H. Muentes, V. A. Domínguez, D. P. Guingla, and E. V. Pinela, "Modelización y Caracterización del Efecto Temperatura en el Cuerpo Humano."
- [13] D. Sánchez Morillo, "Procesado y transmisión de señales biomédicas para el diagnóstico de trastornos y enfermedades del sueño," 2008.
- [14] I. Balas, C. Llumiguano, Z. Horváth, F. Köver, and T. Dóczi, "Talamotomía estereotáxica de la enfermedad de Parkinson y otros tipos de temblor. Experiencias de la actividad multiunitaria burst en el tálamo basada en semimicroelectrodos," *REV NEUROL*, vol. 32, no. 6, pp. 520–524, 2001.
- [15] I. Cifuentes, "Diseño y construcción de un sistema para la detección de señales electromiográficas," *Undergrad. Thesis, Univ. Autónoma Yucatán*, 2010.
- [16] W. auf der Strasse, K. R. G. de Oliveira, L. M. Beraldo, and A. M. W. Stadnik, "SYMMETRIC-ELECTROMYOGRAPHIC ANALYSIS IN THE EVALUATION OF SCOLIOSIS TREATMENT," *Rev. Bras. Med. do Esporte*, vol. 24, pp. 455–459, 2018.
- [17] P. K. Artemiadis and K. J. Kyriakopoulos, "EMG-based control of a robot arm using low-dimensional embeddings," *IEEE Trans. Robot.*, vol. 26, no. 2, pp. 393–398, 2010.
- [18] B. V. F. Vladimir, M. C. M. Javier, A. V. Evgeny, A. Lukyanov, and M. P. L. Emanuel, "Modelado y simulación superior del Robot Mitsubishi RV-

- 2JA controlado mediante señales electromiográficas,” *Enfoque UTE*, vol. 9, pp. 208–222, 2018.
- [19] T. Forbes, “Mouse HCI through combined EMG and IMU,” 2013.
- [20] J. G. Izurieta Freire, “Sistema de adquisición de señales EMG (electromiográficas) para detectar miopatías en deportistas de alto rendimiento,” Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, 2018.
- [21] J. Orozco, E. García, C. Santillán, O. Zambrano, and C. Cevallos, “Prótesis robótica controlada por un brazalete mioeléctrico,” in *Memorias del I Congreso Internacional de Bioingeniería y Sistemas Inteligentes de Rehabilitación*.
- [22] A. Martínez-Miguel, S. A. Vargas-Pérez, E. Gómez-Merlín, M. Arias-Montiel, E. Lugo-González, and R. Miranda-Luna, “Control de Movimiento de una Mano Robótica Mediante Señales Electromiográficas,” in *Memorias del Congreso Nacional de Ingeniería Biomédica*, 2017, vol. 3, no. 1, pp. 73–76.
- [23] E. Koutsos and P. Georgiou, “An analogue instantaneous median frequency tracker for EMG fatigue monitoring,” in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1388–1391.
- [24] F. S. Teles, M. C. Pereira, V. de A. Rocha-Júnior, J. C. do Carmo, and M. M. de Andrade, “Parâmetros eletromiográficos em exercícios fatigantes realizados com diferentes tipos de resistência,” *Fisioter. e Pesqui.*, vol. 23, pp. 257–262, 2016.
- [25] D. Camacho, M. Erazo, E. Mera, and A. Velasco, “Control de un Brazo Robótico LYNX AL5D Empleando Electromiografía a Través de la Tecnología MYO ARMBAND.”
- [26] A. Ganiev, H.-S. Shin, and K.-H. Lee, “Study on virtual control of a robotic arm via a myo armband for the selfmanipulation of a hand amputee,” *Int. J. Appl. Eng. Res.*, vol. 11, no. 2, pp. 775–782, 2016.
- [27] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, and B. Perišić, “Raspberry Pi as Internet of things hardware: performances and constraints,”

Des. issues, vol. 3, no. 8, 2014.

- [28] “LattePanda board - All for Raspberry Pi, Arduino, LattePanda, Orange Pi, Pine64 | by Geekworm.” [Online]. Available: http://www.raspberrypiwiki.com/index.php/LattePanda_board. [Accessed: 09-Aug-2019].
- [29] Y. M. Vasquéz and J. J. Laguardia, “Estudio del flujo vehicular mediante un modelo de Lighthill-Whitham-Richards,” *KnE Eng.*, pp. 449–457, 2018.
- [30] I. M. L. Lagunes, I. G. E. C. A. H. Sánchez, D. R. González, S. S. Cruz, and M. I. I. J. A. Urbano, “Metodología de Análisis del flujo vehicular a través de Simulación Discreta para reducir congestionamientos viales a través de rutas alternas en Misantla Veracruz,” in *Congreso Interdisciplinario de Ingenierías*, 2018, p. 92.
- [31] A. J. Pérez, “Aplicación del algoritmo de Viterbi sobre modelos ocultos de Markov para la estimación de tráfico vehicular,” 2017.
- [32] E. San Juan, M. Jamett, H. Kaschel, and L. Sánchez, “Sistema de reconocimiento de voz mediante wavelets, predicción lineal y redes backpropagation,” *Ingeniare. Rev. Chil. Ing.*, vol. 24, no. 1, pp. 8–17, 2016.
- [33] G. Moreno, E. Abraham, and J. A. Dávalos Pinto, “Reconocimiento de Personas en Ambiente con Emisiones de Humo Usando Sensor Laser y Redes Neuronales Convolucionales desde Nube de Puntos 3D. Parte 1.,” *Bistua Rev. la Fac. Ciencias Básicas*, vol. 17, no. 1, 2019.
- [34] T. C. N. Sandoval, S. V. G. Pérez, F. A. González, R. A. Jaque, and C. Infante, “Uso de redes neuronales artificiales en predicción de morfología mandibular a través de variables craneomaxilares en una vista posteroanterior/Use of Artificial Neural Networks for Mandibular Morphology Prediction through Craniomaxillar Variables...,” *Univ. Odontológica*, vol. 35, no. 74, pp. 21–28, 2016.
- [35] I. Mendoza-Haro and H. Marquetti-Nodarse, “Redes Neuronales Artificiales: factores que determinan la cosecha de caña en la industria azucarera./Artificial Neural Networks: factors that determine the cane harvest in a sugar industry..”

Cienc. Unemi, vol. 12, no. 29, pp. 36–50, 2019.

- [36] F. López-Muñoz, J. Boya, and C. Alamo, “Neuron theory, the cornerstone of neuroscience, on the centenary of the Nobel Prize award to Santiago Ramón y Cajal.,” *Brain Res. Bull.*, vol. 70, no. 4–6, pp. 391–405, Oct. 2006.
- [37] A. Dityatev, D. A. Rusakov, and D. A. Rusakov, “Molecular signals of plasticity at the tetrapartite synapse,” *Curr. Opin. Neurobiol.*, vol. 21, no. 2, pp. 353–359, Apr. 2011.
- [38] P. Larrañaga, “Tema 8. Redes Neuronales.”
- [39] R. D. S. Olarte and C. A. M. Morales, “EL MODELO RED NEURONAL ARTIFICIAL (RNA) COMO ESTRATEGIA INTERDISCIPLINAR PARA EL ACERCAMIENTO A LA COMPRESIÓN DE LOS CIRCUITOS ELÉCTRICOS/ARTIFICIAL NEURAL NETWORK (ANN) MODEL AS AN INTERDISCIPLINARY STRATEGY FOR THE APPROACH TO THE UNDERSTANDING OF ELECTRICAL CIRCUITS,” *Rev. Teckne*, vol. 15, no. 2, 2018.
- [40] E. R. Saldaña, L. Q. Torres, and P. T. Hernández, “Esquema de Aprendizaje Basado en la Ecuación HJB para Redes Neuronales,” 2019.
- [41] Y. B. Rodríguez, “Integración de la red neuronal convolucional con el algoritmo de función de frontera de objeto para reconocimiento de piezas y detección de defectos,” 2018.
- [42] F. L. Saca, A. F. Ramírez, and C. A. Cruz, “PROTOTIPO FUNCIONAL PARA CLASIFICACIÓN DE IMÁGENES CON SALIDA DE AUDIO EN UN SISTEMA EMBEBIDO CON RED NEURONAL CONVOLUCIONAL (FUNCTIONAL PROTOTYPE FOR CLASSIFICATION OF IMAGES WITH AUDIO OUTPUT IN AN EMBEDDED SYSTEM USING CONVOLUTIONAL NEURAL NETWORK),” *Pist. Educ.*, vol. 40, no. 130, 2018.
- [43] J. E. Sierra and M. Santos, “Control de un vehículo cuatrirrotor basado en redes neuronales,” *Actas las XXXVIII Jornadas Automática*, 2017.
- [44] K. G. Molina *et al.*, “Acquisition of Myoelectric Signals: Use of Opensource

Software and Hardware for Signal Preprocessing.," in *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, 2018, pp. 1–5.

ANEXOS

A.1 Arquitectura del proyecto

A.1.1 Funcionamiento y operatividad

El funcionamiento y operatividad del controlador semafórico inteligente puede resumirse en los siguientes pasos:

1. Cuando el agente de tránsito desea tomar el control del tráfico vehicular, existe una opción en el aplicativo desarrollado llamado MODO AGENTE.
2. Al recibir la petición del agente, el control semafórico captura la petición y este ejercerá la función de colocar el semáforo en rojo y encender la luz azul, la cual es el indicador visual de que el semáforo se encuentra en MODO AGENTE.
3. El agente de tránsito, por medio de los movimientos correspondientes a “pare” y “siga”, serán interpretados por la MYO armband y posteriormente enviará dicha señal hasta el controlador principal mediante un dispositivo de transmisión de datos (vía radiofrecuencia).
4. El control semafórico captura la petición interpretando que el semáforo se coloque ya sea en color verde para la señal de “siga” o de color rojo para la señal de “pare”.
5. El agente tiene la potestad y el control de cambio de indicador luminoso en el semáforo, sea este mediante la señal de “pare” o la señal de “siga” para que la MYO armband envíe la señal al dispositivo de transmisión.
6. El control semafórico captura la petición interpretando que el semáforo se coloque en rojo, de acuerdo a la señal generada por el agente de tránsito.



Figura A.1: Funcionamiento y operatividad del control semafórico.

En la figura A.1 se observa el esquemático del funcionamiento y operatividad del control semafórico desarrollado.

A.1.2 Módulo agente

A.1.2.1 Xbee

El Xbee PRO DigiMesh 900 es el modelo de radio que tiene como característica de ser un equipo de bajo costo, pero de alcance de radio frecuencia de largas distancias, trabajando bajo una banda de frecuencia ISM 900 MHz.

Entre las principales características que posee, se encuentran: alto rendimiento y bajo costo (Interior / urbano: hasta 450 pies (140 m), línea de visión exterior: hasta 1,8 millas (3 km), salida de potencia de transmisión: 50 mW (+17 dBm) PIRE, sensibilidad del receptor: -100 dBm, velocidad de datos de RF: 156 kb/s), baja potencia (corriente TX: 210 mA (@ 3.3 V), corriente RX: 80 mA (@ 3.3 V), asíncrono. corriente de reposo: 48 μ A (típico a 3.3 V), corriente de reposo síncrono: 60 μ A (típico a 3.3 V)), redes avanzadas y seguridad (reintentos y agradecimientos, admite punto a punto, punto a multipunto y topologías punto a punto), fácil uso (no se necesita configuración para RF fuera de caja comunicaciones, modos de comando AT y API para configurar el

módulo parámetros, factor de Forma Pequeña, amplio conjunto de comandos, software XCTU gratuito (software de prueba y configuración)).

Para la configuración de los xbee, se requiere de un software gratuito llamado XCTU la cual posee una interfaz gráfica muy amigable y de fácil uso.

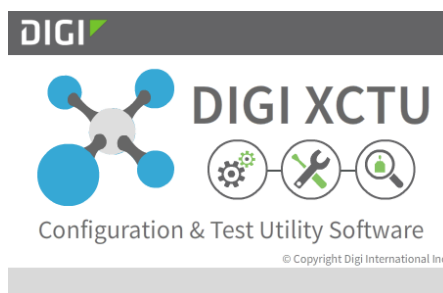


Figura A.2: Software para configuración de un Xbee.

La figura A.2 muestra el logotipo del software utilizado para la configuración de un Xbee.

El Digi XCTU posee herramientas de grandes utilidades las cuales se pueden mencionar: análisis grafica de señales, verificación de nuevas actualizaciones, configuración remota y local, recuperación de fábrica, parámetros de acción, entre los principales.

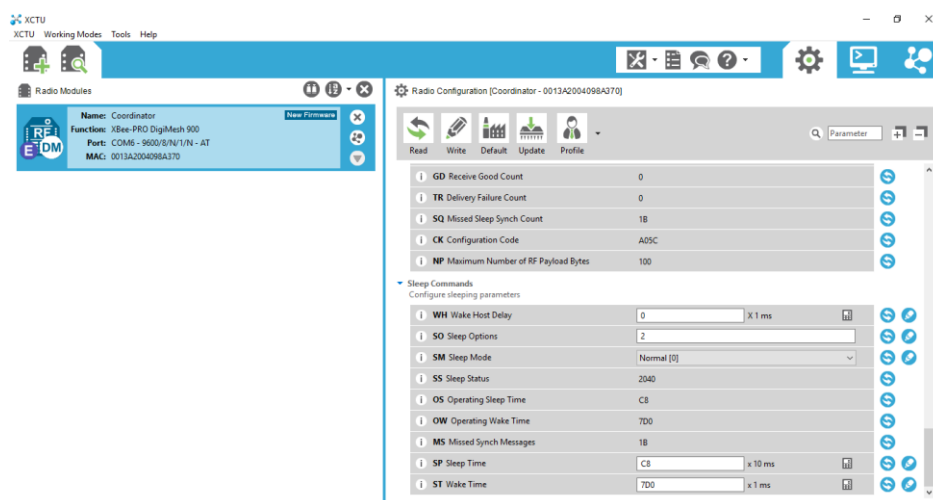


Figura A.3: Entorno del software Digi XCTU.

En la figura A.3 se muestra el entorno del software Digi XCTU para la configuración y parametrización de un Xbee.

A.1.2.2 Raspberry

El dispositivo Raspberry tiene una gran cantidad de modelos la cual ya fueron explicados en este documento, pero para este proyecto se utilizó el modelo Raspberry Pi 3 modelo B+, ya que contiene las características necesarias para la implementación, como son la cantidad de puertos USB disponibles, los cuales son utilizados para una memoria externa, mouse y teclado.

Para el funcionamiento del sistema desarrollado, se escoge el sistema operativo raspbian junto con el IDE Thonny Python.

El IDE Thonny Python es una herramienta de programación para desarrollo en Python. Python es un lenguaje de programación de bajo nivel que permite realizar desarrollos sin costo de memoria alto y, con las herramientas de pygame, nos permite desarrollar el entorno gráfico que se requería.

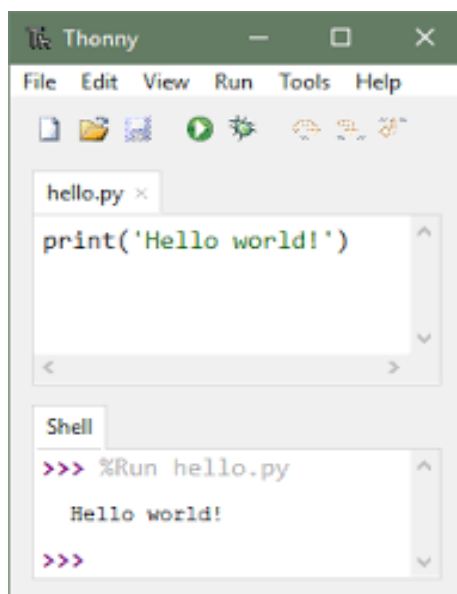


Figura A.4: Entorno de programación de Thonny Python.

En la figura A.4 se presenta el entorno de programación del software de Thonny Python.

Para el reconocimiento del dispositivo MYO en el ambiente de raspbian, es necesario realizar la descarga de componentes importantes para un buen funcionamiento y manejo del dispositivo. A continuación, se presentan los comandos a ser ejecutados:

```
// plug bluetooth adapter
// permission to ttyACM0 - must restart linux user after this
sudo usermod -a -G dialout $USER

// dependencies
sudo apt-get install python-pip
sudo pip install pySerial --upgrade
sudo pip install enum34
sudo pip install PyUserInput
sudo apt-get install python-Xlib
sudo apt-get install python-tk

// now reboot
```

Estos comandos hacen referencia a las interfases que utiliza la MYO: puerto serial, bluetooth y librerías de MYO para Python.

Luego de realizar la instalación de los componentes, se requiere el visor manager de MYO más conocido como PyoConnect, el cual es una interfaz que permite realizar la conexión del dispositivo Raspberry mediante bluetooth con el dispositivo MYO. PyoConnect se encuentra en la versión 2.0 siendo la más actualizada.

Dentro del paquete del PyoConnect, se encuentra el archivo PyoManager.pyc, el cual, al ser ejecutado, se invoca la interfaz gráfica de configuración del MYO, tal como se muestra en la figura A.5.



Figura A.5: Interfaz gráfica de PyoConnect para configuración de MYO.

Su funcionamiento permite realizar la conexión y desconexión del dispositivo, así como también permitir si el dispositivo MYO tiene el control del mouse, video, etc.

Dentro de la programación, Se utilizaron componentes adicionales para la interface grafica como es pygame e interface de comunicación como pyserial y las librerías descargadas para manipular el dispositivo myo como son common.py, myo_raw.py etc.

A continuación, se adjunta las líneas de código del sistema:

Nombre programa: **TrafficLightData_Agente.py**

```
#!/usr/bin/env python
from __future__ import print_function
import time
import pygame
import sys,os
import myo_raw
import serial

from pygame import *
from pygame.locals import *

b=(255,255,255)
contador = 0
lista = []

class Cursor (pygame.Rect):
    def __init__(self):
        pygame.Rect.__init__(self,0,0,1,1)
    def update (self):
        self.left, self.top=pygame.mouse.get_pos()

class Boton (pygame.sprite.Sprite):
```

```

def __init__ (self,b1,b2,x=200,y=200):
    self.imagen_normal = b1
    self.imagen_seleccion = b2
    self.imagen_actual = self.imagen_normal
    self.rect = self.imagen_actual.get_rect()
    self.rect.left,self.rect.top=(x,y)
def update (self,pantalla,cursor):
    if cursor.collidect (self.rect):
        self.imagen_actual=self.imagen_seleccion
    else: self.imagen_actual = self.imagen_normal

    pantalla.blit (self.imagen_actual, self.rect)

def proc_emg(emg, moving, times=[]):
    if GetContador() == 1:
        lista.clear()
        #print(GetContador())

lista.append((str(emg[0])+", "+str(emg[1])+", "+str(emg[2])+", "+str(emg[3])+", "+str(emg
[4])+", "+str(emg[5])+", "+str(emg[6])+", "+str(emg[7]) + "|"))
    #if contador == 90:
        #try:
            #tempDatosEMG = "<data>" + datosEMG + "</data>"
            #EnviarPuertoSerial(tempDatosEMG)
            #contador = 0
            #tempDatosEMG = ""
            #datosEMG = ""
        #except:
            #contador = 0
            #tempDatosEMG = ""
            #datosEMG = ""
def DefaultVar():

```

```
global contador
contador = 0

def DefaultLista():
    global lista
    lista.clear()

def GetContador():
    global contador
    contador = contador + 1
    return contador

def main():

    pygame.init()
    ventana = pygame.display.set_mode((300,430))
    pygame.display.set_caption("Myo Control Semafórico")
    myCursor = Cursor()

    ventana.fill(b)
    pygame.display.update()

    datosEMG = ""
    MYO = 0
    CONECTADO = 0
    DESCONECTADO = 0
    MODO_AGENTE = 0
    MODO_NORMAL = 0
    MODO_SOLICITUD = 0
    PUERTO_SERIAL = 0

    try:
```

```

m = myo_raw.MyoRaw(sys.argv[1] if len(sys.argv) >= 2 else None)
m.add_emg_handler(proc_emg)
MostrarMensajeOk("Se encontro el dispositivo de conexión MYO", ventana, 35)
MYO = 1
except:
    MostrarMensajeError("No se encontro el dispositivo de conexión MYO", ventana,
25)

if MYO == 1:
    try:
        ser = serial.Serial(
            port='/dev/ttyS0',
            baudrate = 9600
        )
        PUERTO_SERIAL = 1
    except:
        MostrarMensajeError("Error en la configuración del puerto serial", ventana, 40)

img_conectar =
pygame.image.load("/home/pi/Pictures/img_conectar.png").convert_alpha()
img_conectar2 =
pygame.image.load("/home/pi/Pictures/img_conectar2.png").convert_alpha()

img_normal =
pygame.image.load("/home/pi/Pictures/img_normal.png").convert_alpha()
img_normal2 =
pygame.image.load("/home/pi/Pictures/img_normal2.png").convert_alpha()

img_agente =
pygame.image.load("/home/pi/Pictures/img_agente.png").convert_alpha()
img_agente2 =
pygame.image.load("/home/pi/Pictures/img_agente2.png").convert_alpha()

```

```

    img_desconectar                                     =
pygame.image.load("/home/pi/Pictures/img_desconectar.png").convert_alpha()
    img_desconectar2                                   =
pygame.image.load("/home/pi/Pictures/img_desconectar2.png").convert_alpha()

    img_salir = pygame.image.load("/home/pi/Pictures/img_salir.png").convert_alpha()
    img_salir2                                     =
pygame.image.load("/home/pi/Pictures/img_salir2.png").convert_alpha()

    img_solicitud                                     =
pygame.image.load("/home/pi/Pictures/img_solicitud.png").convert_alpha()
    img_solicitud2                                   =
pygame.image.load("/home/pi/Pictures/img_solicitud2.png").convert_alpha()

    botonConectar = Boton(img_conectar,img_conectar2,40,30)
    botonAgente = Boton(img_agente,img_agente2,40,90)
    botonSolicitud = Boton(img_solicitud,img_solicitud2,40,150)
    botonNormal = Boton(img_normal,img_normal2,40,210)
    botonDesconectar = Boton(img_desconectar,img_desconectar2,40,270)
    botonSalir = Boton(img_salir,img_salir2,40,330)

    capturar = 1

    while True:
        pygame.display.update()
        #ventana.fill(b)
        eventos = pygame.event.get()

        for evento in eventos:
            if evento.type == QUIT:
                pygame.quit()

```

```

sys.exit()
if evento.type==pygame.MOUSEBUTTONDOWN:
    if myCursor.collidirect(botonConectar.rect):
        if evento.button == 1:
            try:
                if MYO == 1:
                    if PUERTO_SERIAL == 1:
                        CONECTADO = 1
                        DESCONECTADO = 0
                        MODO_NORMAL = 1
                        MostrarMensajeOk("Conexión MYO exitosa", ventana, 80)
                    else:
                        MostrarMensajeError("Error en la configuración del puerto
serial", ventana, 40)
                else:
                    MostrarMensajeError("No se encontro el dispositivo de conexión
MYO", ventana, 25)
            except:
                MostrarMensajeError("Error al conectarse al MYO", ventana, 80)
    if myCursor.collidirect(botonAgente.rect):
        if evento.button == 1:
            try:
                if MYO == 1:
                    if CONECTADO == 1:
                        EnviarPuertoSerial('<data>CMD002</data>')
                        MODO_AGENTE = 1
                        MODO_NORMAL = 0
                        MostrarMensajeOk("MODO AGENTE exitoso", ventana, 80)
                    else:
                        MostrarMensajeError("MYO no conectado", ventana, 80)
                else:
                    MostrarMensajeError("No se encontro el dispositivo de conexión

```



```

MYO", ventana, 25)
    except:
        MostrarMensajeError("Error MODO AGENTE", ventana, 80)
if myCursor.colliderect(botonSolicitud.rect):
    if evento.button == 1:
        #try:
            if MYO == 1:
                if CONECTADO == 1:
                    if MODO_AGENTE == 1:
                        DefaultLista()
                        MODO_SOLICITUD = 1
                        pygame.mixer.music.load('referee.mp3')
                        pygame.mixer.music.play(0)
                        m.connect()
                        pygame.time.delay(1500)
                        while (True):
                            #GetContador()
                            if len(lista) >= 100:
                                break
                            m.run()
                            #print(contador)
                            # print(len(lista))
                            for i in range(0, len(lista)):
                                datosEMG = datosEMG + lista[i]
                            #print(datosEMG)
                            EnviarPuertoSerial("<data>" + datosEMG + "</data>")
                            m.disconnect()
                            DefaultVar()
                            DefaultLista()
                            #print(len(datosEMG))
                            datosEMG = ""
                        pygame.mixer.music.load('referee.mp3')

```

```

        pygame.mixer.music.play(0)
    else:
        MostrarMensajeError("Debe encontrarse en MODO
AGENTE", ventana, 55)
    else:
        MostrarMensajeError("MYO no conectado", ventana, 80)
    else:
        MostrarMensajeError("No se encontro el dispositivo de conexión
MYO", ventana, 25)
    #except:
        #MostrarMensajeError("Error MODO SOLICITUD", ventana, 80)
    if myCursor.colliderect(botonNormal.rect):
        if evento.button == 1:
            try:
                if PUERTO_SERIAL == 1:
                    MODO_AGENTE = 0
                    MODO_NORMAL = 1
                    EnviarPuertoSerial('<data>CMD001</data>')
                    MostrarMensajeOk("MODO NORMAL exitoso", ventana, 80)
                else:
                    MostrarMensajeError("Error en la configuración del puerto serial",
ventana, 40)
            except:
                MostrarMensajeError("Error MODO NORMAL", ventana, 80)
    if myCursor.colliderect(botonDesconectar.rect):
        if evento.button == 1:
            try:
                if MYO == 1:
                    CONECTADO = 0
                    DESCONECTADO = 1
                    MODO_AGENTE = 0
                    MODO_NORMAL = 0

```

```

        MostrarMensajeOk("Desconexión MYO exitosa", ventana, 75)
    else:
        MostrarMensajeError("No se encontro el dispositivo de conexión
MYO", ventana, 25)
    except:
        MostrarMensajeError("Error al desconectarse al MYO", ventana, 75)
if myCursor.colliderect(botonSalir.rect):
    if evento.button == 1:
        print("Click boton salir")
        pygame.quit()
        sys.exit()

    botonConectar.update(ventana,myCursor)
    botonAgente.update(ventana,myCursor)
    botonNormal.update(ventana,myCursor)
    botonDesconectar.update(ventana,myCursor)
    botonSolicitud.update(ventana,myCursor)
    botonSalir.update(ventana,myCursor)
    myCursor.update()

def MostrarMensajeError(mensaje, ventana, posicion):
    ventana.fill(b)
    myFont = pygame.font.Font(None, 16)
    label = myFont.render(mensaje,0,(248,0,0))
    ventana.blit(label,(posicion,400))
    pygame.display.update()

def MostrarMensajeOk(mensaje, ventana, posicion):
    ventana.fill(b)
    myFont = pygame.font.Font(None, 16)
    label = myFont.render(mensaje,0,(84, 153, 199))
    ventana.blit(label,(posicion,400))

```

```

pygame.display.update()

def EnviarPuertoSerial(mensaje):
    ser = serial.Serial(port='/dev/ttyS0',baudrate = 9600)

    ser.write(mensaje.encode())
    ser.close()

main()

```

Las opciones que fueron creadas para el aplicativo del módulo de agente son las que se muestran en la figura A.6 a continuación:



Figura A.6: Interfaz del aplicativo del módulo de agente.

A continuación, una breve descripción de los botones que componen la interfaz del módulo de agente:

- Conectar: Esta opción permite realizar la conexión con el dispositivo MYO; si todo es correcto, aparecerá un mensaje indicando el éxito, caso contrario indica un error en la conexión.
- Modo Agente: Esta opción permite indicar que el semáforo debe pasar a modo

agente, es decir, que permita aceptar las solicitudes de movimientos que se realizan en la opción “*Solicitud*”; además, se inicia la modalidad mediante el encendido de los colores rojo y azul.

- **Solicitud:** Esta opción permite enviar los movimientos que se realizan por medio de la MYO durante un tiempo determinado. Para indicar la duración del lapso mencionado, se notificará por medio de un sonido tanto el inicio como el final. Este sonido es similar al pito que maneja un agente de tránsito.
- **Modo Normal:** Esta opción indica que el semáforo pasará a modo normal, es decir, que no permita aceptar las solicitudes de movimientos que se realizan en la opción “*Solicitud*”. Se manifiesta la activación de dicha modalidad mediante el apagado del indicador azul en el semáforo.
- **Desconectar:** Esta opción permite realizar la desconexión con el dispositivo MYO; si todo es correcto, aparecerá un mensaje indicando el éxito, caso contrario indica un error en la desconexión.
- **Salir:** Permite salir del aplicativo.

A.1.2.3 Lattepanda

El dispositivo de Lattepanda tiene una gran cantidad de modelos la cual ya fueron explicados en este documento, pero para este proyecto se utilizará el modelo 2GB/32GB.

Este modelo posee 2 GB RAM y 32 GB de disco duro y sistema operativo Windows 10 con Arduino Leonardo.

Para la recepción de la información emitida por el dispositivo del agente, es necesario colocar un Xbee en el puerto serial disponible.

Para el funcionamiento del sistema desarrollado, se utiliza el sistema operativo Windows 10 con los siguientes aplicativos: Visual Studio 2017, MatLab R2019a, Arduino Studio.

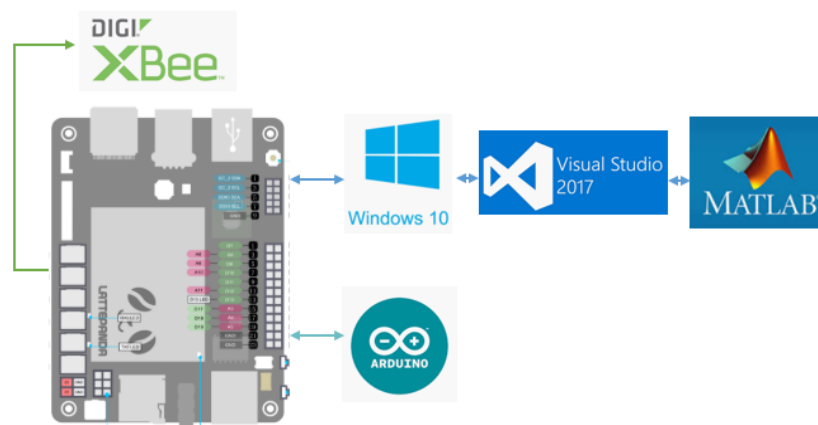


Figura A.7: LattePanda y la gama de software para el desarrollo aplicativo.

En la figura A.7 se observa la tarjeta del LattePanda y los distintos softwares que fueron instalados para el desarrollo del aplicativo.

Dentro de la instalación del Matlab, se debe agregar herramientas adicionales que serán importantes para el funcionamiento del sistema. Dentro de esta gama de herramientas, se encuentra la Deep Learning Toolbox, la cual proporciona un marco para diseñar e implementar redes neuronales profundas con algoritmos, modelos entrenados y aplicaciones. Puede utilizar redes neuronales convolucionales (ConvNets, CNN) y redes de memoria a largo plazo (LSTM) para realizar la clasificación y la regresión en imágenes, series temporales y datos de texto.

A continuación, se adjunta la programación correspondiente:

Nombre programa: TrafficLightData.EMG.sln

TrafficLightData.EMG.Lib

Command.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace TrafficLightData.EMG.Lib
{
    public class Command
    {
        public const string MODO_NORMAL = "CMD001";
        public const string MODO_AGENTE = "CMD002";
        public const string SEMAFOROS_ROJO = "CMD003";
        public const string SEMAFOROS_VERDE = "CMD004";
        public const string SEMAFOROS_AMARILLO = "CMD005";
        public const string SEMAFORO_ROJO = "CMD006";
        public const string SEMAFORO_VERDE = "CMD007";
        public const string SEMAFOROS_AMARILLO_INTERMITENTE = "CMD008";
    }
}
```

Logger.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;

namespace TrafficLightData.EMG.Lib
{
    public class Logger
    {
        private string p_RutaLog = string.Empty;
        private string p_NombreLog = string.Empty;
        private static object myObj;
```

```

public Logger(string nombreLog)
{
    this.p_RutaLog =
System.IO.Path.Combine(Assembly.GetExecutingAssembly().Location.Substring(0,
Assembly.GetExecutingAssembly().Location.LastIndexOf("\\"), "Log");
    this.p_NombreLog = nombreLog;

    myObj = new object();
    try
    {
        if (!Directory.Exists(p_RutaLog))
            Directory.CreateDirectory(p_RutaLog);
    }
    catch (Exception)
    {}
}

public void WriteLine(Exception ex, int estado = 2)
{
    this.WriteLine(ex.Message, estado);
}

public void WriteLine(string mensajeError, int estado = 2)
{
    string[] data = new string[1];
    data[0] = mensajeError;

    DateTime today = DateTime.Now;
    StringBuilder sb = new StringBuilder("[ " + today.ToString("yyyy-MM-dd
HH:mm:ss.fff") + " ]");
    for (int i = 0; i < data.Length - 1; i++)

```



```

        sb.Append("[ " + data[i] + " ]");
sb.Append("-" + data[data.Length - 1]);

lock (myObj)
{
    StreamWriter sw = null;
    try
    {
        sw = new StreamWriter(p_RutaLog.TrimEnd("\\") + "\\ " + p_NombreLog +
        "_" + today.ToString("yyyyMMdd") + ".txt", true);
        sw.WriteLine(sb.ToString());
        if (estado == 0)
            Console.ForegroundColor = ConsoleColor.White;
        if (estado == 1)
            Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine(sb.ToString());
        Console.ResetColor();
    }
    catch (Exception)
    {
    }
    finally
    {
        if (sw != null)
        {
            sw.Flush();
            sw.Dispose();
        }
        sw = null;
    }
}
}

```

```

    }
}

```

MatlabFunction.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TrafficLightData.EMG.Lib
{
    public class MatlabFunction
    {
        private MLab.MLab p_Matlab = null;

        public MatlabFunction()
        {
            this.p_Matlab = new MLab.MLab();
        }

        public int ObtenerEstadoSemaforo(string rutaFuncion, string nombreArchivo)
        {
            string rutaArchivoFuncionMatlab = rutaFuncion;

            this.p_Matlab.Execute(@"cd " + rutaArchivoFuncionMatlab);

            object result = null;

            this.p_Matlab.Feval("traffilightv02", 1, out result, nombreArchivo);
            object[] res = result as object[];
            return int.Parse(res[0].ToString());
        }
    }
}

```

```

    }
}
}

```

Transaction.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Configuration;
using System.IO;
using System.IO.Ports;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Timers;

namespace TrafficLightData.EMG.Lib
{
    public class Transaction
    {
        private string p_RutaArchivosCSV = "";

        private SerialPort p_SerialPortAgente = null;
        private SerialPort p_SerialPortSemaforo = null;

        private string p_ResultadoAgente = "";
        private string p_ResultadoSemaforo = "";

        private Timer p_CheckTimer = null;
        private Logger p_Logger = null;
        private AppConfiguracion p_Config = null;
    }
}

```

```

private List<string> p_ListaAgente = new List<string>();
private List<string> p_ListaArchivos = new List<string>();

private MatlabFunction p_MatlabFunction = null;

public Transaction(Logger logger)
{
    this.p_Logger = logger;
    this.p_MatlabFunction = new MatlabFunction();
}

public void Do()
{
    try
    {
        this.p_ResultadoAgente = "";
        this.p_ResultadoSemaforo = "";

        this.p_Config = new AppConfiguracion();

        this.p_RutaArchivosCSV = this.p_Config.DirectoríoEjecucionArchivo;

        string puertoAgente = "COM" + this.p_Config.PuertoComAgente;
        string puertoSemaforo = "COM" + this.p_Config.PuertoComSemaforo;

        this.p_SerialPortAgente = new SerialPort(puertoAgente);
        this.p_SerialPortSemaforo = new SerialPort(puertoSemaforo);

        this.p_SerialPortAgente.DataReceived +=
SerialPortAgente_DataReceived;
        this.p_SerialPortSemaforo.DataReceived +=
SerialPortSemaforo_DataReceived;

```

```
this.p_CheckTimer = new Timer(this.p_Config.TiempoEjecucionArchivo);
this.p_CheckTimer.Elapsed += CheckTimer_Elapsed;
this.p_CheckTimer.Start();

if (!this.p_SerialPortAgente.IsOpen)
    this.p_SerialPortAgente.Open();

if (!this.p_SerialPortSemaforo.IsOpen)
    this.p_SerialPortSemaforo.Open();

}
catch (Exception ex)
{
    this.EscribirLog("TRANSACTION_DO", ex.Message);

    try
    {
        if (this.p_SerialPortAgente != null)
            if (this.p_SerialPortAgente.IsOpen)
                this.p_SerialPortAgente.Close();

        if (this.p_SerialPortSemaforo != null)
            if (this.p_SerialPortSemaforo.IsOpen)
                this.p_SerialPortSemaforo.Close();

        if (this.p_CheckTimer != null)
            this.p_CheckTimer.Stop();
    }
    catch (Exception ex1)
    {
```

```

        this.EscribirLog("TRANSACTION_DO", ex1.Message);
    }
}

private void CheckTimer_Elapsed(object sender, ElapsedEventArgs e)
{
    try
    {
        this.p_CheckTimer.Stop();

        string[] archivos = ObtenerArchivosCSV();

        foreach (string archivo in archivos)
        {
            if (!this.p_ListaArchivos.Exists(x => x == archivo))
            {
                this.p_ListaArchivos.Add(archivo);
                //this.EscribirLog("TRANSACTION_CHECKTIMER", "ARCHIVO: " +
archivo);

                System.Threading.Thread thread = new System.Threading.Thread(()
=> ThreadMainProcesoMatlab(archivo));
                thread.Start();
            }
        }
    }
    catch (Exception ex)
    {
        this.EscribirLog("TRANSACTION_CHECKTIMER", ex.Message);
    }
    finally
    {

```

```

        this.p_CheckTimer.Start();
    }
}

private void ThreadMainProcesoMatlab(string nombreArchivo)
{
    int estado = this.ObtenerEstadoRNMatLab(nombreArchivo);
    if (estado <= 1)
    {
        this.EscribirLog("TRANSACTION_CHECKTIMER", "ESTADO: " +
estado.ToString(), estado);
        //System.Threading.Thread.Sleep(1000);
    }
    string comando = "";
    switch (estado)
    {
        case 0:
            comando = Command.SEMAFORO_ROJO;
            break;
        case 1:
            comando = Command.SEMAFORO_VERDE;
            break;
        case 2:
            comando = Command.SEMAFOROS_AMARILLO_INTERMITENTE;
            break;
    }

    if (estado <= 1)
        this.EnviarComandosSemaforo(comando);
    this.EliminarArchivoCSV(nombreArchivo);
}

```

```

private int ObtenerEstadoRNMatLab(string nombreArchivo)
{
    try
    {
        return
this.p_MatlabFunction.ObtenerEstadoSemaforo(this.p_Config.DirectorioEjecucionAr
chivo, nombreArchivo);
    }
    catch (Exception ex)
    {
        this.EscribirLog("TRANSACTION_OBTENERESTADORNMATLAB",
ex.Message);
        return 3;
    }
}

private bool EnviarComandosSemaforo(string comando)
{
    bool bResultado = false;

    try
    {
        byte[] buffer = Encoding.ASCII.GetBytes(comando);
        this.p_SerialPortSemaforo.Write(buffer, 0, buffer.Length);
        bResultado = true;
    }
    catch (Exception ex)
    {
        this.EscribirLog("TRANSACTION_ENVIARCOMANDOSSEMAFORO",
ex.Message);
    }
}

```



```

        return bResultado;
    }

    private void SerialPortSemaforo_DataReceived(object sender,
SerialDataReceivedEventArgs e)
    {
        try
        {
            this.p_ResultadoSemaforo = this.p_ResultadoSemaforo +
this.p_SerialPortSemaforo.ReadExisting();
        }
        catch (Exception ex)
        {

this.EscribirLog("TRANSACTION_SERIALPORTSEMAFORO_DATARECEIVED",
ex.Message);
        }
    }

    private void SerialPortAgente_DataReceived(object sender,
SerialDataReceivedEventArgs e)
    {
        try
        {

            this.p_ResultadoAgente = this.p_ResultadoAgente +
this.p_SerialPortAgente.ReadExisting();
            //System.Threading.Thread.Sleep(3000);
            //Console.WriteLine(">> " + this.p_ResultadoAgente);

            while (true)

```

```

{
    if (!this.p_ResultadoAgente.Contains("<data>"))
        break;
    if (!this.p_ResultadoAgente.Contains("</data>"))
        break;
    int index0 = this.p_ResultadoAgente.IndexOf("<data>");
    int index1 = this.p_ResultadoAgente.IndexOf("</data>");
    this.p_ResultadoAgente = this.p_ResultadoAgente.Substring(index0);
    int index2 = (index1 + 7) - index0;
    string nuevaLinea = this.p_ResultadoAgente.Substring(0, index2);
    this.p_ListaAgente.Add(nuevaLinea);
    this.p_ResultadoAgente = this.p_ResultadoAgente.Substring(index2);
}
foreach (string data in this.p_ListaAgente)
{
    string linea = data.Replace("<data>", "").Replace("</data>", "");
    if (!linea.Equals("CMD001") && !linea.Equals("CMD002"))
    {
        string[] datos = linea.Split('|');
        this.CrearArchivoCSV(datos);
        //System.Threading.Thread thread = new System.Threading.Thread(()
=> ThreadMainProcesoMatlab(archivo));
        //thread.Start();
    }
    else
    {
        this.EnviaComandosSemaforo(linea);
        Console.WriteLine(linea);
    }
}
this.p_ListaAgente.Clear();

```

```

    }
    catch (Exception ex)
    {

this.EscribirLog("TRANSACTION_SERIALPORTAGENTE_DATARECEIVED",
ex.Message);
    }
}

private string CrearArchivoCSV(string[] datos)
{
    string nombreArchivo = DateTime.Now.ToString("yyyyMMddHHmmss") +
".csv";
    var rutaNombreArchivo = Path.Combine(this.p_RutaArchivosCSV,
nombreArchivo);
    File.WriteAllLines(rutaNombreArchivo, datos);
    return nombreArchivo;
}

private string[] ObtenerArchivosCSV()
{
    return Directory.GetFiles(this.p_RutaArchivosCSV, "*.csv");
}

private void EliminarArchivoCSV(string nombreArchivo)
{
    string ruta = Path.Combine(this.p_RutaArchivosCSV, nombreArchivo);
    if (File.Exists(ruta))
        File.Delete(ruta);
}

private void EscribirLog(string funcion, string mensajeError, int estado = 2)

```

```

    {
        string mensaje = "{" + funcion + "}" + mensajeError;
        this.p_Logger.WriteLine(mensaje, estado);
    }

    public void Dispose()
    {
        try
        {
            if (this.p_SerialPortAgente.IsOpen)
                this.p_SerialPortAgente.Close();

            if (this.p_SerialPortSemaforo.IsOpen)
                this.p_SerialPortSemaforo.Close();

            if (this.p_SerialPortAgente != null)
                this.p_SerialPortAgente.DataReceived -=
SerialPortAgente_DataReceived;

            if (this.p_SerialPortSemaforo != null)
                this.p_SerialPortSemaforo.DataReceived -=
SerialPortSemaforo_DataReceived;
        }
        catch (Exception ex)
        {
            this.EscribirLog("TRANSACTION_DISPOSE", ex.Message);
        }
    }
}

class AppConfiguracion
{

```

```

public string PuertoComAgente { get; private set; }
public string PuertoComSemaforo { get; private set; }
public int TiempoEjecucionArchivo { get; private set; }
public string DirectorioEjecucionArchivo { get; private set; }

public AppConfiguracion()
{
    string ruta =
System.IO.Path.Combine(Assembly.GetExecutingAssembly().Location.Substring(0,
Assembly.GetExecutingAssembly().Location.LastIndexOf("\\")),
"TrafficLightData.EMG.Lib.dll.config");
    var map = new ExeConfigurationFileMap { ExeConfigFilename = ruta };
    Configuration configFile =
ConfigurationManager.OpenMappedExeConfiguration(map,
ConfigurationUserLevel.None);
    this.PuertoComAgente =
configFile.AppSettings.Settings["COM_AGENTE"].Value.ToString();
    this.PuertoComSemaforo =
configFile.AppSettings.Settings["COM_SEMAFORO"].Value.ToString();
    this.TiempoEjecucionArchivo =
int.Parse(configFile.AppSettings.Settings["EJEC_ARCHIVO_MS"].Value.ToString());
    this.DirectorioEjecucionArchivo =
configFile.AppSettings.Settings["EJEC_ARCHIVO_DIR"].Value.ToString();
}
}
}

```